Search-Based Software Engineering: Trends, Techniques and Applications

MARK HARMAN, University College London
S. AFSHIN MANSOURI, Brunel University
YUANYUAN ZHANG, University College London

In the past five years there has been a dramatic increase in work on Search-Based Software Engineering (SBSE), an approach to Software Engineering (SE) in which Search-Based Optimization (SBO) algorithms are used to address problems in SE. SBSE has been applied to problems throughout the SE lifecycle, from requirements and project planning to maintenance and reengineering. The approach is attractive because it offers a suite of adaptive automated and semiautomated solutions in situations typified by large complex problem spaces with multiple competing and conflicting objectives.

This article[1] provides a review and classification of literature on SBSE. The work identifies research trends and relationships between the techniques applied and the applications to which they have been applied and highlights gaps in the literature and avenues for further research.

Categories and Subject Descriptors: D.2 [**Software Engineering**]

General Terms: Algorithms, Design, Experimentation, Management, Performance

Additional Key Words and Phrases: Software engineering, search-based techniques, survey

ACM Reference Format:
Hurman, M., Mansouri, S. A., and Zhang, Y. 2012. Search-Based software engineering: Trends, techniques and applications. ACM Comput. Surv. 45, 1, Article 11 (November 2012), 61 pages.
DOI = 10.1145/2379776.2379787 http://doi.acm.org/10.1145/2379776.2379787

1. INTRODUCTION

Software Engineering (SE) often considers problems that involve finding a suitable balance between competing and potentially conflicting goals. There is often a bewilderingly large set of choices and finding good solutions can be hard. For instance, the following is an illustrative list of SE questions.

(1) What is the smallest set of test cases that covers all branches in this program?
(2) What is the best way to structure the architecture of this system to enhance its maintainability?
(3) What is the set of requirements that balances software development cost and customer satisfaction?

The article is a (significantly) extended version of the recent ICSE "The current state and future of search based software engineering" paper by M. Harman, one of the present authors [Harman 2007b].

The authors are very grateful for the support of the collaborators of the EPSRC-funded project SEBASE and the EU-funded project Evo Test. Funding from these two large projects (EP/D050863 and IST-33472 respectively) provided part financial support for the work undertaken in the production of this article.

Authors' addresses: M. Harman, Department of Computer Science, University College London, UK; S. A. Mansouri, Brunel Business School, Brunel University, UK; Y. Zhang (corresponding author), Department of Computer Science, University College London, UK; email: yuanyuan.zhang@cs.ucl.ac.uk.

DOI 10.1145/2379776.2379787 http://doi.acm.org/10.1145/2379776.2379787

(4) What is the best allocation of resources to this software development project?
(5) What is the best sequence of refactoring steps to apply to this system?

Answers to these questions might be expected from literature on testing, design, requirements engineering, SE management, and refactoring, respectively. It might appear that these questions, which involve different aspects of software engineering, would be covered by different conferences and specialized journals and would have little in common. However, all of these questions are essentially *optimization* questions. As such, they are typical of the kinds of problem for which SBSE is well adapted and with which each has been successfully formulated as a search-based optimization problem. As we shall see in this survey, SBSE has been applied to testing, design, requirements, project management, and refactoring. This survey will show that work on SBSE applied to each of these five areas addresses each of the five questions raised before. This breadth of applicability is one of the enduring appeals of SBSE.

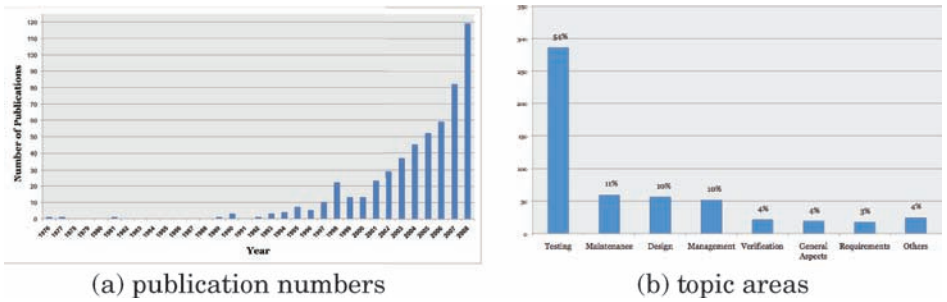
In SBSE, the term "search" is used to refer to the metaheuristic Search-Based Optimization (SBO) techniques that are used. SBSE seeks to reformulate SE problems as SBO problems (or "search problems" for short). The use of the term "search" should not to be confused with "search" from other contexts such as textual or hypertextual search. Rather, for SBSE, a search problem is one in which optimal or near-optimal solutions are sought in a search space of candidate solutions, guided by a fitness function that distinguishes between better and worse solutions.

The interest in SBO for SE has led to an increased interest in other forms of optimization for SE that are not necessarily directly based on a "search". In the literature it is common to find the term "SBSE" applied to any form of optimization in which the problem domain comes from SE and the solution involves optimization according to some well-defined notion of fitness. In this article, we therefore include classical Operations Research (OR) techniques as well as metaheuristic "search-based" techniques in our understanding of SBSE.

It has been argued that the virtual nature of software makes it well suited for SBO [Harman 2010]. This is because fitness is computed directly in terms of the engineering artifact, without the need for the simulation and modeling inherent in all other approaches to engineering optimization. The field of SE is also imbued with rich metrics that can be useful initial candidates for fitness functions [Harman and Clark 2004]. This article aims to provide a comprehensive survey of SBSE. It presents research activity in categories drawn from the ACM subject categories within SE. For each, it lists the papers, drawing out common themes, such as the type of search technique used, the fitness definitions, and the nature of evaluation.

A wide range of different optimization and search techniques can and have been used. The most widely used are local search, Simulated Annealing (SA), Genetic Algorithms (GAs), Genetic Programming (GP), and Hill Climbing (HC). There is also increasing evidence of industrial interest in SBSE, with uptake by many software-centric organizations including Daimler [Bühler and Wegener 2008; Harman et al. 2007a; Wegener et al. 2001; Windisch et al. 2007], Ericsson [Zhang et al. 2010], IBM [Yoo et al. 2009, 2011a], Microsoft [Lakhotia et al. 2010; Xie et al. 2008], Motorola [Baker et al. 2006], Nokia [Del Rosso 2006], and NASA [Feather et al. 2004].

As the article reveals, 54% of the overall SBSE literature is concerned with SE applications relating to testing. There have been several important surveys in this widely studied general area [Afzal et al. 2009; Ali et al. 2010; McMinn 2004]. For this reason, the present survey will report overall trends in the wider SBSE literature (including Search-Based Testing), but it will defer to these other three surveys for details on the specific subfield of Search-Based Testing. The reader is also referred to



(a) publication numbers (b) topic areas

Fig. 1. The trend of publications on SBSE and Software Engineering topic area.

an earlier (but considerably longer) version of this article [Harman et al. 2009] that contains a detailed section on testing.

There has been a considerable increase in the quantity of SBSE research over the past few years (see Figure 1(a)). Despite the excellent work in the surveys listed earlier, there remains, to date, no comprehensive survey of the whole field of study concerning trends in research. It is therefore timely to review the SBSE literature, the relationships between the applications to which it has been applied, the techniques used, trends, and open problems.

The primary contributions of this survey are as follows.

(1) *Coverage and completeness*. The survey gathers publication data and trends, covering SBSE from its early origins to a publication "census date" of December 31st 2008. This census date is chosen for pragmatic reasons. As this survey reveals, there is a notably increasing trend of publication in SBSE. The growth in activity in this area makes a survey useful, but it also means that it may not be feasible to conduct a detailed survey after this date.
(2) *Classification*. The classification of SE areas allows us to identify gaps in the literature, indicating possible areas of SE that could (but have yet to) benefit from the application of SBSE. Similarly, the analysis of search techniques used allows us to identify SBO algorithms that have yet to receive significant attention. We also apply Formal Concept Analysis (FCA) [Snelting 1998] in order to explore the relationships between techniques and the applications to which they have been applied.
(3) *Trend analysis*. The survey presents numeric data concerning trends which give a quantitative assessment of the growth in the area and the distributions of activity among the SE domains that have received attention. We are also able to identify recent growth areas.

2. BACKGROUND

Although interest in SBSE has witnessed a recent dramatic rise, its origins can be traced back to early work on optimization in SE in the 1970s. The earliest currently known attempt to apply optimization to an SE problem was reported by Miller and Spooner [1976] in 1976 in the area of software testing. The term SBSE was first used by Harman and Jones [2001a] in 2001. This paper acted as a "manifesto" for SBSE, but it should also be noted that much earlier, Carl Chang had also used his IEEE Software editorial to promote the more widespread use of evolutionary computation in SE in 1994 [Chang 1994].

Table I. The Classification Scheme for SBSE Literature

Classification Criteria	Values
Type of activity (ACM coding)	Network Protocols (C.2.2), Requirements/Specifications (D.2.1), Design Tools and Techniques (D.2.2), Coding Tools and Techniques (D.2.3), Software/Program Verification (D.2.4), Testing and Debugging (D.2.5), Distribution, Maintenance and Enhancement (D.2.7), Management (D.2.9), Distributed Artificial Intelligence (I.2.11), Security and Protection (K.6.5)
Objectives (or fitness)	Maximum Cohesion, Minimum Coupling, ...
Representation method	Tree, Graph, String, *etc.*
Search techniques	Greedy Search, HC, GAs, SA, Tabu Search (TS), Other Search Techniques
Problems used for evaluation	Real World Data, Synthetic Data

Figure 1(a) provides a histogram charting SBSE publication growth over time, while Figure 1(b) shows the proportion of papers that fall into each of the different SE application area subject categories.

Harman and Jones [Harman 2007b; Harman and Jones 2001a] identified two key ingredients for the application of SBO to SE problems:

(1) the choice of the representation of the problem; and
(2) the definition of the fitness function.

This simplicity and ready applicability makes SBSE a very attractive option. Typically, a software engineer will have a suitable representation for her problem, because one cannot do much engineering without a way to represent the problem in hand. Furthermore, many problems in SE have a rich and varied set of software metrics associated with them that naturally form good initial candidates for fitness functions [Harman and Clark 2004]. With these two ingredients it becomes possible to implement SBO algorithms.

Naturally, there is a lot more to the application of these techniques, but these two simple ingredients are sufficient to get started with experimentation. Poulding et al. [2007] presented a framework for experimental investigation of the different algorithms. An overview of search techniques is available in other surveys [Harman 2007b], while a more detailed treatment of search methodologies can be found in the book edited by Burke and Kendall [2005].

3. CLASSIFICATION SCHEME

Our classification of SE activities is taken from the Association for Computing Machinery (ACM) Computing Classification System, projected onto those SE areas to which SBSE has been applied (see Table I). A list of query keywords was constructed for each of the activities and each of the search techniques (see Table II). For example, the search term used to locate papers on Search-Based Requirements/Specifications (D.2.1) was:

((requirements OR specifications OR next release OR release planning OR requirements selection OR requirements analysis OR COTS OR requirements prioritization OR requirements triage) AND (search based OR optimization OR multiobjective optimization OR search techniques))

We used the following sources from which to search: Google Scholar, IEEE Xplore Digital Library, ACM Digital Library, SpringerLink, ScienceDirect, and Wiley

Table II. Search Terms

Publication Classification	Search Terms
Network Protocols	protocol OR message exchange OR communication
Requirements/Specifications	requirements OR specifications OR next release OR release planning OR requirements selection OR requirements analysis OR COTS OR requirements prioritisation OR requirements triage
Design Tools and Techniques	software design OR design quality OR design pattern OR software architecture OR QoS OR component integration OR cohesion OR coupling OR synthesis OR fault tolerance OR OO design
Coding Tools and Techniques	program slices OR grammar inference
Software/Program Verification	model checking OR verification OR synthesis
Distribution, Maintenance and Enhancement	maintenance OR refactoring OR modularization OR evolution OR real time OR quality prediction OR legacy systems OR migration
Management	project planning OR project management OR scheduling OR staffing OR cost estimation OR effort estimation
Distributed Artificial Intelligence	agent OR multiagent
Security and Protection	security OR immune system OR AIS

AND

Search Techniques	Search Terms
	search based OR optimization OR multiobjective optimization OR genetic algorithms OR GAs OR genetic programming OR GP OR hill climbing OR simulated annealing OR local search OR Integer programming OR ant colony optimization OR ACO OR PSO OR EDA

InterScience. We also asked the researchers in the field to check the references and notify us of the missing references.

4. REQUIREMENTS/SPECIFICATIONS

Requirements engineering is a vital part of the SE process [Cheng and Atlee 2007], to which SBSE has also been applied in order to optimize choices among requirements, the prioritization of requirements, and the relationships between requirements and implementations.

Bagnall et al. [2001] suggested the term Next Release Problem (NRP) for requirements release planning and described various metaheuristic optimization algorithms, including greedy algorithms, branch-and-bound, SA, and HC. The authors did not give any *value* property to each requirement. They only used an associated *cost*. The task of the work was to find a subset of stakeholders whose requirements are to be satisfied. The objective was to maximize the cumulative measure of the stakeholder's importance to the company under resource constraints. This single-objective formulation-based NRP was the first attempt on SBSE for requirements.

Feather and Menzies [2002] built an iterative model to seek the near-optimal attainment of requirements. The authors proposed a Defect Detection and Prevention (DDP) process based on a real-world instance: a NASA pilot study. The DDP combined the requirements interaction model with the summarization tool to provide and navigate the near-optimal solutions in the risk mitigation/cost trade-off space. The paper was one of the first to use Pareto optimality in SBSE for requirements. The Pareto fronts were not produced using multiobjective optimization techniques (as with more recent work by Jalali et al. [2008]), but were produced using the iterative application of a

weighting-based single-objective formulation by applying SA. Also, with relevance to Pareto-optimal formulations, Feather et al. [2006, 2004] summarized the visualization techniques used to present requirements status, including Pareto front plotted by SA.

Ruhe et al. [Greer and Ruhe 2004; Ruhe and Greer 2003; Ruhe and Ngo-The 2004] proposed the GA-based approaches known as the EVOLVE family that aimed to maximize the benefits of delivering requirements in an incremental software release planning process. Their approaches balance the required and available resources; assessing and optimizing the extent to which the ordering conflicts with stakeholder priorities. They also took requirement changes and two types of requirements interaction relationship into account and provided candidate solutions for the next release in an iterative manner. As with previous work, the authors use a single-objective formulation, taking the resource budget as a constraint.

Moreover, Carlshamre [2002] took requirements interdependencies into consideration by using Linear Programming (LP) techniques. Ruhe and Saliu [2005] also presented an Integer Linear Programming (ILP)-based method that combined computational intelligence and human negotiation to resolve their conflicting objectives. Van den Akker et al. [Li et al. 2007; van den Akker et al. 2005] further extended the technique and developed an optimization tool based on ILP, integrating the requirements selection and scheduling for the release planning to find the optimal set of requirements with the maximum revenue to cater for budgetary constraints.

Using search-based techniques to choose components to include in different releases of a system was studied by Harman et al. [Baker et al. 2006; Harman et al. 2006]. Baker et al. [2006] addressed the problem of determining the next set of releases of a software via ranking and selection of candidate software components. They use greedy and SA algorithms. Harman et al. [2006] also considered requirements problems as a feature subset selection problems, presenting results on a single-objective formulation for a real-world dataset from Motorola.

The work of AlBourae et al. [2006] was focused more on the requirements change handling, that is, replanning of the product release. A greedy replan algorithm was adopted to reduce risks and increase the number of requirements achieved in the search space under change.

In addition, Cortellessa et al. [2006, 2008b] described an optimization framework to provide decision support for Code Off-The-Shelf (COTS) and in-house components selection. The ILP LINGO model solver optimization models (CODER, DEER) were proposed to automatically satisfy the requirements while minimizing the cost.

Like many problems in SE, such as project planning, NRP, and regression testing, there is a relationship between feature subset selection problems and feature ordering (prioritization) problems. A comparison of approaches (both analytical and evolutionary) for prioritizing software requirements was proposed by Karlsson et al. [1998]. Greer and Ruhe [2004] also provided a method for optimally allocating requirements to increments, based on:

(1) a means of assessing and optimizing the degree to which the ordering conflicts with stakeholder priorities within technical precedence constraints;
(2) a means of balancing required and available resources for all increments;
(3) an overall method for continuous planning of incremental software development based on a GA.

The aforementioned work on this problem has tended to treat the requirements selection and optimization as a single-objective problem formulation, in which the various constraints and objectives that characterize the requirements analysis problem are combined into a single-objective fitness function. Single-objective formulations have the drawback that the maximization of one concern may be achieved at the expense

of the potential maximization of another, resulting in a bias guiding the search to a certain part of the solution space.

Zhang et al. [2007b] provided a multiobjective formulation of the NRP to optimize value and cost. They present the results of an empirical study into the suitability of multiobjective search techniques.

Early work on integration by Saliu and Ruhe [2007] showed how implementation objectives and requirements objectives could be simultaneously optimized using a multiobjective optimization approach. Like Zhang et al. [2007b], this work also formulated the problem as a two-objective Pareto-optimal problem, but in this case with implementation-level and requirement-level objectives, where as Zhang et al. [2007b] use cost and value as their two objectives.

Finkelstein et al. [2008] showed how a multiobjective optimization approach can be used to explore fairness of outcomes in requirements assignments. There are different definitions of fairness. For example, each customer might wish to receive equal spend from the developers, or they might prefer that they receive an equal number of their desired requirements compared to other customers. Finkelstein et al. show how these different definitions of fairness can be considered to be different objectives to be optimized.

The application of SBSE optimization techniques to requirements analysis problems provides one example of an SE application that is often regarded as inherently imprecise, qualitative, and informal. However, using SBSE it can be formalized as a quantitative multiobjective optimization problem. A position paper on recent trends in requirements analysis optimization was provided by Zhang et al. [2008].

5. DESIGN TOOLS AND TECHNIQUES

In other engineering disciplines SBO is widely used as a means of developing better designs. Where there are widely accepted metrics, such as cohesion and coupling, there has been much work on optimizing these [Doval et al. 1999; Harman et al. 2002, 2005; Mahdavi et al. 2003b; Mancoridis et al. 1999, 1998; Mitchell and Mancoridis 2002, 2003, 2008; Mitchell et al. 2002, 2004]. However, this previous work on cohesion and coupling is not concerned with design per se. Rather, it is concerned with the problem of reconstructing the module boundaries of a system after implementation. As such, this previous work is categorized as work on maintenance, rather than work on design in this survey. Räihä [2010] provided a recent detailed survey of SBSE techniques for both design problems and redesign (maintenance) problems in SE.

Clearly, there is a relationship between redesign (for software maintenance) and design (as a part of the initial design phase of the lifecycle). This relationship is borne out naturally in the literature on software design, where some of the SBSE techniques from software maintenance also have been adapted for software design. Simons and Parmee [2006, 2007, 2008a, 2008b] proposed multiobjective GAs to address Object-Oriented (OO) software design. Like the previous work on cohesion and coupling for software maintenance [Harman et al. 2002, 2005; Mancoridis et al. 1999, 1998; Mitchell and Mancoridis 2003, 2008] the fitness function is inspired by similar SE goals. However, the goal is upstream software design rather than more downstream maintenance. O'Keeffe and Ó Cinnéide [2003, 2004] converted OO software design to an optimization problem using SA. A set of metrics is used for evaluating the design quality. This is a development of work by the same authors on refactoring OO systems according to metrics (that is described in Section 7.2).

It would be natural to suppose that work on design patterns [Gamma et al. 1995] could and should form a foundation for a strand of work on SBSE for design. This possibility has recently been explored in detail by [Räihä 2008a, 2008b; Räihä et al. 2008a],

who proposed a GA-based approach to automatically synthesize software architectures consisting of several design patterns.

Other authors have proposed new SBSE approaches, specifically targeted at the design phase of the software development process. Feldt [1999] presented a model to explore the difficulty in early software development phases by using GP and also describes a prototype of interactive software development workbench called WISE that uses biomimetic algorithms [Feldt 2002]. Several authors have also considered SBSE techniques for balancing Quality-of-Service (QoS) objectives, such as Khoshgoftaar et al. [2004a, 2004b], who proposed an approach for calibrating a multiobjective Module-Order Model (MOM) using GP.

The problem of QoS-aware Web service composition was introduced by Canfora et al. [2005a], who use GAs to solve QoS-aware composition optimization problems. This problem, which lies at the heart of service-oriented computing, implemented as Web-based systems, has recently been taken up and studied by other authors. Jaeger and Mühl [2007] discussed the Quality-of-Service-based Web services selection problems using GAs. Ma and Zhang [2008] proposed a GA-based method for Web service composition and Web service selection that takes into account QoS constraints. Zhang et al. [Su et al. 2007; Zhang et al. 2006, 2007a] applied GAs for Web services selection with global QoS constraints.

Several authors have addressed the design problem of component selection and integration. This component selection problem is closely related to the requirement assignment problem. Baker et al. [2006] presented results on greedy optimization and SA for component selection, while Yang et al. [2006] proposed an approach for the software integration problem by using GAs to reduce risk. Classical OR techniques have also been applied to component selection problems: Desnos et al. [2008] combined backtracking and branch-and-bound techniques for the automatic component substitution problem to optimize software reuse and evolution. Other authors have considered the component selection problem as a selection optimization problem. For example, Cortellessa et al. [2008a] presented a framework to support the selection of COTS components. These approaches minimize system construction cost. Vijayalakshmi et al. [2008] proposed a GA-based approach to select an optimized combination of components and Kuperberg et al. [2008] proposed a GP-based platform-independent reengineered parametric behavior model for black-box components performance prediction.

State-based models of design are increasingly popular and these create opportunities for SBSE research because of the wealth of research on synthesis of state-based models from examples, using optimization techniques. Goldsby et al. [Goldsby and Cheng 2008a, 2008b; Goldsby et al. 2008] presented an evolution-based tool for software behavioral model generation to improve the quality of systems. The system, Avida-MDE, generates a set of communicating state-based models of system behavior using model inference techniques that allow a finite state machine model to be synthesized from cases. A related approach was used by Lucas and Reynolds [2005], who presented an EA for learning deterministic finite automaton to optimally assign state labels and compare its performance with the evidence-driven state merging algorithm.

Feldt, one of the early pioneers of the application of SBO to SE, showed how fault tolerance could be designed into systems using GP to evolve multiple diverse software variants [Feldt 1998a, 1998b, 1998c]. This is a novel approach to N-version computing, in which highly fault-tolerant systems are created several times, in different ways, to increase robustness. The goal was to increase quality since the GP evolved versions would be qualitatively different from any human-generated "diverse versions".

In the traditional N-version computing approach, different teams of programmers are deployed to develop the different (and hopefully, therefore, diverse) solutions to the same problem. Of course, the development of different versions of a system in this

manner is a highly expensive solution to the problem of robustness and fault tolerance; it can and has only been used in highly safety-critical situations, where the expense might be justified. Though it was not directly the intention of the work, Feldt's work also showed that by using GP to evolve the required diverse solutions to the same problem, there is the potential to use SBSE techniques to overcome the expense that was previously inherent in *N*-version computing.

Work on SBSE techniques for design has grown in prevalence in the last three years, with many new and interesting SE design problems emerging. Amoui et al. [2006] applied GAs to optimize OO metrics and find the best sequence of system transformations in order to improve the quality of the transformed design. This approach shares some similarities with work on refactoring using SBO to find good sequences of refactoring steps. Barlas and El-Fakih [2008] presented a GA-based method for mapping client-server problems to optimize the delivery of applications to multiple clients by multiple servers. Bowman et al. [2008] applied the Strength Pareto Evolutionary Algorithm 2 (SPEA2) multiobjective optimization algorithm to provide-decision support system for the Class Responsibility Assignment (CRA) problem. Cao et al. [2005b] addressed the cost-driven Web service selection problem by using GAs. Chardigny et al. [2008b] proposed a search-based approach to the extraction of component-based architectures of OO systems. As with other work in this section, this work could be categorized as design or as redesign, highlighting the interplay in SE between design, maintenance, and evolution of software systems. Sharma and Jalote [2008] proposed a heuristic approach for deploying software components that maximizes performance.

6. SOFTWARE/PROGRAM VERIFICATION AND MODEL CHECKING

Model checking is an area of research that could well benefit from more research on SBSE techniques, because model checking throws up enormous search spaces and there are candidate metrics to guide a search. Software/Program Verification (ACM: D.2.4) is given in Table VIII. Godefroid was the first to apply SBO to explore the state space used in model checking [Godefroid 1997]. Where the state space is too large to be fully checked, search-based optimization can be used to identify isomorphic subgraphs and to seek out counterexamples. Alba et al. [Alba and Chicano 2007a, 2007b, 2007c; Alba et al. 2008; Chicano and Alba 2008a, 2008b, 2008c] also showed how Ant Colony Optimization (ACO) can be used to explore the state space used in model checking to seek counterexamples. Mahanti and Banerjee [2006] also proposed an approach for model checking, using ACO and Particle Swarm Optimization (PSO) techniques.

Other authors have also explored the relationship between SBSE and model checking. For instance, Johnson [2007] used model checking to measure fitness in the evolution of finite state machines, while Katz and Peled [2008a, 2008b] provided a model-checking-based GP approach for verification and synthesis from specification. They present an approach that combines Hoare-logic-style assertion-based specifications and model checking within a GP framework [He et al. 2008].

7. DISTRIBUTION, MAINTENANCE AND ENHANCEMENT

Software maintenance is the process of enhancing and optimizing deployed software (software release), as well as remedying defects. It involves changes to the software in order to correct defects and deficiencies found during field usage as well as the addition of new functionality to improve the software's usability and applicability.

Much of the work on the application of SBSE to these topics has tended to focus on two strands of research, each of which has attracted a great deal of interest and around which a body of work has been produced.

The first topic to be addressed is search-based software modularization. More recently, there have also been several developments in search-based approaches to

the automation of refactoring. The previous work on distribution, maintenance, and enhancement is discussed in more detail in the following two subsections, which separately consider work on modularization and refactoring.

Other work on SBSE application in distribution, maintenance, and enhancement that does not fall into these two categories has considered the evolution of programming languages [Van Belle and Ackley 2002], real-time task allocation[2] [Bate and Emberson 2006; Emberson and Bate 2007], quality prediction based on the classification of metrics by a GA [Vivanco and Pizzi 2004], and legacy systems migration [Sahraoui et al. 2002]. SBSE has also been applied to the concept assignment problem. Gold et al. [2006] applied GAs and HC to find overlapping concept assignments. Traditional techniques (which do not use SBSE) cannot handle overlapping concept boundaries, because the space of possible assignments grows too rapidly. The formulation of this problem as an SBSE problem allows this large space to be tamed.

7.1. Modularization

Mancoridis et al. were the first to address the problem of software modularization using SBSE [Mancoridis et al. 1998] in 1998. Their initial work on HC for clustering modules to maximize cohesion and minimize coupling was developed over the period from 1998 to 2008 [Doval et al. 1999; Mancoridis et al. 1999; Mitchell and Mancoridis 2002, 2003, 2008; Mitchell et al. 2002, 2004]. The pioneering work of Macoridis et al. led to the development of a tool called Bunch [Mancoridis et al. 1999] that implements software module clustering.

The problem of module clustering is similar to the problem of finding near cliques in a graph, the nodes of which denote modules and the edges of which denote dependence between modules. Mancoridis et al. [1999] called this graph a module dependency graph. The Bunch tool produces a hierarchical clustering of the graph, allowing the user to select the granularity of cluster size that best suits his application.

Following Macoridis et al., other authors also developed the idea of module clustering as a problem within the domain of SBSE. Harman et al. [2002] studied the effect of assigning a particular modularization granularity as part of the fitness function, while Mahdavi et al. [Mahdavi et al. 2003b; Mahdavi 2005] showed that combining the results from multiple hill climbs can improve on the results for simple HC and GAs. Harman et al. also [Harman et al. 2005] explored the robustness of the Modularization Quality (MQ) fitness function in comparison with an alternative measure of cohesion and coupling, EValuation Metric (EVM), used in work on clustering gene expression data.

Other authors have also considered search-based clustering problems. Bodhuin et al. [2007b] applied GAs to group together class clusters in order to reduce packaging size and the average downloading times. Huynh and Cai [2007] applied GAs to cluster Design Structure Matrices and check the consistency between design and source code structures.

Despite several attempts to improve on the basic HC approach [Harman et al. 2002; Mahdavi et al. 2003b; Mitchell and Mancoridis 2002], this simple search technique has been found very effective for this problem. However, Praditwong et al. [2010] recently demonstrated that multiobjective optimization can significantly outperform HC in terms of modularization quality. Mitchell and Mancoridis recently published a survey of the Bunch project and related work [Mitchell and Mancoridis 2006].

[2]This work could equally well be categorized as "real-time SBSE", a topic area which is sure to develop in the future, given the highly constrained nature of the real-time environment and the many competing objectives that have to be optimized.

Clustering is a very general problem to which a number of algorithms have been applied, not merely search-based algorithms. Clustering is likely to find further applications in SE applications, beyond the original work on software modular structure. For example, Cohen et al. [2006] showed how search-based clustering algorithms could be applied to the problem of heap allocation Java program optimization. Memory fragmentation issues have also been addressed using SBSE: Del Rosso [2006] improved internal memory fragmentation by finding the optimal configuration of a segregated free lists data structure using GAs.

7.2. Refactoring

In refactoring work, the goal is to change the program, altering its structure without altering the semantics. Closely related topics have also been addressed. For example, Reformat et al. [2003, 2007] explored applications of software clones and present a method for automatic clone generation using GP. Clones are also a focus for attention in the work of Di Penta et al. [Di Penta et al. 2005; Di Penta 2005], who proposed a language-independent software renovation framework to remove unused objects, clones, and circular dependencies, and to cluster large libraries into more cohesive and smaller ones. Cowan et al. [2004] provided a framework of automatic programming applying GP. Bouktif et al. [2006a] used SBSE techniques to schedule refactoring actions in order to remove duplicated code. Antoniol et al. [2003] proposed a GA-based refactoring process to reduce size and minimize coupling of libraries. Bodhuin et al. [2007a] introduced a tool to support refactoring decisions using a GA guided by software metrics.

Search-based refactoring work can be partitioned according to whether the goal is to optimize the program to a refactored version of itself [Cooper et al. 1999; O'Keeffe and Ó Cinnéide 2006, 2007, 2008a, 2008b; Ryan 2000; Williams 1998] or whether it is to optimize the sequence of refactoring steps to be applied [Harman and Tratt 2007; Williams 1998]. The work can also be categorized according to whether the approach followed is single objective (combining all metrics into a single fitness value) [O'Keeffe and Ó Cinnéide 2006, 2007, 2008a, 2008b; Ryan 2000; Seng et al. 2005, 2006; Williams 1998] or multiobjective (using Pareto optimality to separately optimize each metric) [Harman and Tratt 2007]. Bouktif et al. [2006a] proposed an approach to schedule refactoring actions under constraints and priorities in order to remove duplicated code.

This work is closely related to that on statement-level search-based transformation, which was first explored by Ryan and Williams in the context of identification of transformations that improve imperative language paralellizability [Ryan 2000; Williams 1998]. Nisbet [1998] also applied a GA to determine the optimal transformation sequence that minimizes the execution time of FORTRAN programs for Single Program Multiple Data (SPMD) execution on parallel architectures.

Stephenson et al. [2003b] used GP to improve compiler heuristics. This approach directly evolves the heuristic deployed by the compiler. Hoste and Eeckhout [2008] and Dubach et al. [2007] used an alternative approach to improve the performance of compiled code by searching the space of compiler options that control optimization levels in `gcc`. There are about 60 such flags (purely for optimization behavior of the `gcc`), making for a nontrivial search space of options, specifically targeted at performance of the compiled code. The two objectives considered in the paper are compilation time and code quality (in terms of execution time), though many other possibilities suggest themselves, such as the many nonfunctional properties of the program being compiled.

Refactoring seeks to restructure a program to improve some aspect of the structure without affecting the behavior of the restructured system. It is an example of a more general approach: (source-to-source) program transformation, to which SBSE

techniques have also been applied. Fatiregun et al. [2003, 2005, 2004] and Kessentini et al. [2008] applied transformations to reduce programs size and to automatically construct amorphous slices. The first authors that have considered any form of source-to-source transformation using a search-based approach were Cooper et al. [1999] who applied search-based transformation to find sequences of compiler optimizations. This work used only whole program transformations. The work of Ryan, Williams [2000], and Fatiregun et al. [1998], which followed focused on the more "microlevel" or statement-level transformations.

By contrast with this previous work on transformation, the work on refactoring is more concerned with the OO paradigm, but the principles used in the refactoring work are largely the same as those that pertain to the statement-level transformation domain.

8. MANAGEMENT

SE management is concerned with the management of complex activities being carried out in different stages of the software lifecycle, seeking to optimize both the processes of software production as well as the products produced by this process. Task and resource allocation, scheduling, and cost-effort estimation have been among the most frequently considered problems studied in this category. Papers on SBSE for management can be roughly categorized according to whether they concern *project planning* activities or whether they create predictive models for *cost estimation* to provide decision support to software project managers. The following two subsections present the work in each of these two categories.

8.1. Project Planning

Chang et al. [Chang 1994; Chang et al. 1994, 1998, 2001; Chao et al. 1993] were the first to use SBSE on software management problems. Their early work on search-based software project management [Chang 1994; Chang et al. 1994; Chao et al. 1993] introduced the Software Project Management Net (SPMNet) approach for project scheduling and resource allocation, evaluating SPMNet on simulated project data. SPMNet deals with project scheduling and resource allocation. Other early work on SBSE for project management was presented by Aguilar-Ruiz et al. [2001, 2002], who also advocated the use of a Software Project Simulator (SPS) to evaluate fitness, guiding an evolutionary search for a set of management rules to inform and assist the project manager.

The allocation of teams to work packages in software project planing can be thought of as an application of a bin packing problem [Coffman et al. 1984]. Motivated by this observation, Antoniol et al. [2004a, 2005], Chicano and Alba [Alba and Chicano 2005, 2007d] applied search algorithms to software projects. Antoniol et al. applied GAs, HC, and SA to the problem of staff allocation to work packages. They also considered problems of reworking and abandonment of projects, which are clearly important aspects of most SE projects. Antoniol et al. applied the algorithms to real-world data from a large Year 2000 (Y2K) maintenance project. Chicano and Alba considered the multiobjective version of the problem applied to synthetic data. The multiple objectives are combined into a single fitness function using weights for each of the component objectives.

Bouktif et al. have used SBSE to consider the management problem of determining the expected quality of a software system as a prediction system. Bouktif et al. [2006a, 2002] presented a GA-based quality model to improve software quality prediction, while Bouktif et al. [2006b] showed how the general problem of combining quality experts, modeled as Bayesian classifiers, can be tackled via an SA algorithm customization. Bouktif et al. [2004] used a GA-based method to improve rule-set-based OO software quality prediction.

The application areas of software project management, scheduling, and planning have witnessed a great deal of recent interest from the research community, with recent contributions from a number of authors. Alvarez-Valdés et al. [2006] used a Scatter Search (SS) algorithm for project scheduling problems to minimize project completion duration. This is one of the few applications of SS in SBSE. Barreto et al. [2008] proposed an optimization-based project staffing algorithm to solve the staffing problem. Cortellessa et al. [2008b] described an optimization framework to provide decision support for software architects. Hericko et al. [2008] used a simple gradient-based optimization method to optimize project team size while minimizing project effort. Kapur et al. [2008] used a GA to provide optimal staffing for product release and best quality to customers under time constraints. Kiper et al. [2007] applied GAs and SA to select an optimal subset of Verification and Validation (V&V) activities in order to reduce risk under budget restrictions, thereby linking the problem domains of testing and management.

It is clear that this application area will continue to draw interest and activity from the SBSE community. Though there has been much interest in the difficulty of the problem of software project management, there remain a number of unresolved challenges, including the following.

(1) *Robustness*. It may not be sufficient to find a project plan that leads to early completion time. It may be more important to find plans that are robust in the presence of changes. Such a robust plan may be suboptimal with respect to the completion time objective. This may be a worthwhile sacrifice for greater certainty in the worst-case completion time, should circumstances change. These forms of "robustness trade-off" have been widely studied in the optimization literature [Beyer and Sendhoff 2007].
(2) *Poor Estimates*. All work on software project estimation has had to contend with the problem of notoriously poor estimates [Shepperd 2007]. Much of the work on SBSE for project management has implicitly assumed that reliable estimates are available at the start of the project planning phase. This is an unrealistic assumption. More work is required in order to develop techniques for software project planning that are able to handle situations in which estimates are only partly reliable.
(3) *Integration*. Software project management is a top-level activity in the software development lifecycle. It draws in other activities such as design, development, testing, and maintenance. As such, project management is ideally not an activity that can be optimized in isolation. In order to achieve wider applicability for the SBSE approach to software project management, it will be necessary to develop techniques that can integrate management activities with these other engineering activities.
Software project management also cannot be conducted in isolation from requirement engineering, since the choice of requirements may affect the feasibility of plans. Therefore, though the requirements gathering and analysis phases typically precede the formulation of management planning, this is clearly not desirable once one accepts that the planning phase can be formulated as an optimization problem. Early work on integration by Saliu and Ruhe [2007] showed how implementation objectives and requirements objectives could be simultaneously optimized using a multiobjective optimization approach. More work is required to integrate other aspects of the software development process into an optimized software project management activity.

Figure 2 provides a generic schematic overview of SBSE approaches to project planning. Essentially, the approach is guided by a simulation that captures, in abstract form, the conduct of the project for a given plan. A project plan is evaluated for fitness

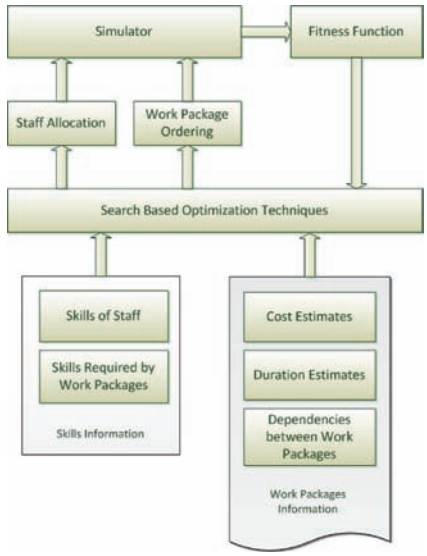


Fig. 2. A generic search-based project management scheme.

using the simulation. Typically the simulation is a simple queuing simulation that can deterministically compute properties of the project (such as completion time), based upon the plan. The plan involves two aspects: people and tasks. The tasks (usually called work packages) have to be completed by teams. There may be dependencies between the work packages which mean that one cannot start until another is completed. Work packages may also require certain skills, possessed by some staff (and not others), while staff may be assigned to teams.

These details form the basis of the different choices of formulation of the problem studied in the literature. However, all are united by the overall approach, which is to assess fitness of a project plan, using a model of its conduct, with the search space of possible project plans. They all take into account different aspects of the real-world software project management problem as determined by the problem formulation.

8.2. Cost Estimation

Software project cost estimation is known to be a very demanding task [Shepperd 2007]. For all forms of project, not merely those involving software, project estimation activities are hard problems, because of the inability to predict the unpredictable and the natural tendency to allocate either arbitrary (or zero) cost to unforeseen (and unforeseeable) necessitated activities. The problem of estimation is arguably more acute for software projects than it is for projects in general, because of:

(1) the inherent uncertainties involved in software development;
(2) the comparative youth of the SE as a discipline; and
(3) the wide variety of disparate tasks to which SE solutions can be applied.

Dolado was the first author to attack software project estimation problems using SBSE. He applied GP to the problem of cost estimation, using a form of "symbolic regression" [Dolado 2000, 2001; Dolado and Fernandez 1998]. The idea was to breed simple mathematical functions that fit the observed data for project effort (measured in function points). This has the advantage that the result is not merely a prediction system, but also a function that *explains* the behavior of the prediction system.

Several authors have used GP in cost estimation and quality prediction systems. Evett et al. [1999] used GP for quality prediction. Liu and Khoshgoftaar [2001] also applied GP to quality prediction, presenting two case studies of the approach. This GP approach has been extended, refined, and further explored by Khoshgoftaar et al. [Khoshgoftaar and Liu 2007; Khoshgoftaar et al. 2003, 2008; Liu and Khoshgoftaar 2004, 2003]. In all these works, GP evolved predictors are used as the basis for decision support. Other authors have used a combination of GA and GP techniques for estimation as a decision support tool for software managers. Huang et al. [2008] integrated the grey relational analysis with a GA to improve the accuracy of software effort estimation. Jarillo et al. [2001] applied GAs and GP to effort estimation for predicting the number of defects and estimating the reliability of the system. Lokan [2005] investigated the performance of GP-based software effort estimation models using a number of fitness functions.

Burgess and Lefley also reported results from the application of GP to software project cost estimation [Burgess and Lefley 2001; Lefley and Shepperd 2003]. Shan et al. [2002] compared a grammar-guided GP approach with linear regression in estimation of software development cost. Sheta [2006] presented two new model structures to estimate the effort required for the development of software projects using GAs and benchmarked them on a NASA software project dataset. Shukla [2000] presented a neuro-genetic approach using a genetically trained Neural Network (NN) predictor trained to predict resource requirements for a software project based on historical data.

Kirsopp et al. [2002] also used search techniques in software project cost estimation. Their approach predicts unknown project attributes in terms of known project attributes by seeking a set of near-neighbor projects that share similar values for the known attributes. This approach is known as Case-Based Reasoning (CBR) and it is widely used in prediction systems. CBR works well when the existing base of project data is of consistently good quality, but can perform badly where some projects and/or attributes are miss-recorded. Kirsopp et al. [2002] showed that the problem of determining a set of good predictors can be formulated as a feature subset selection problem, to which they applied an HC algorithm. This work was also one of the few in the SBSE literature that has evaluated the properties of the search landscape.

9. ANALYSIS OF TECHNIQUES & APPLICATIONS

Figure 1(a) showed the trend of growth in publications in SBSE, while Figure 1(b) showed how the application areas within SE have been covered. In this section a further and deeper analysis of the overall area is provided using bar graphs to show the relative frequency of application of optimization techniques, together with a formal concept lattice to show the relationships between application areas and techniques applied.

Figure 3 shows the distributions of search-based optimization techniques used in SBSE. Perhaps one striking aspect of the SBSE literature (from the optimization point of view) is the comparatively widespread use of HC. This simple local search technique is often derided in the optimization literature, yet it can be effective and has a number of advantages over more sophisticated algorithms.

(1) It is efficient: both quick to implement and fast in execution.
(2) Though it may become trapped in a local optima, it can be restated multiple times. As such, for problems in which a quick answer is required that is merely "good enough", a solution which is sufficiently better than the current one so that the effort in adopting it would offset the effort, HC often serves the purpose; the choice of other techniques may denote something of a "sledge hammer to crack a nut".

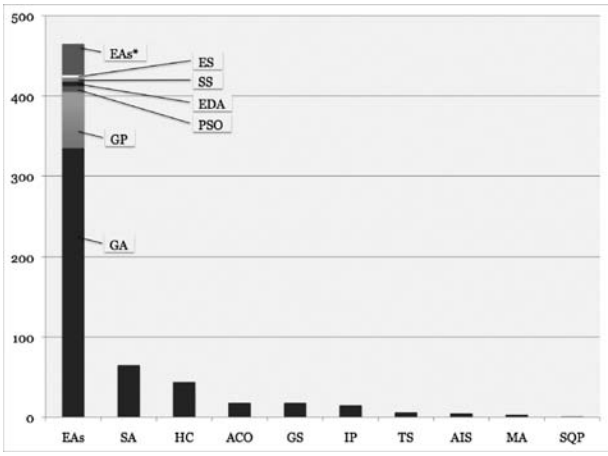


Fig. 3. Numbers of papers using each of the different types of SBO techniques: EAs are split into GA, GP, ES, SS, EDA, PSO, and EAs. In this figure the stacked bar "EAs" represents the general class of all Evolutionary Algorithms, while the top portion of bar labeled "EAs*" refers to the proportion of the literature that describes itself as using an "evolutionary algorithm", without further qualification or specification as to the type of evolutionary algorithm used.

(3) It gives a sense of the landscape structure. Because HC performs a local search and ascends the "nearest hill to the start point", with multiple restarts, it can be a quick and effective way of obtaining a first approximation to the structure of the landscape.

These properties of HC make it well suited to new application areas of SBSE (or indeed for any new optimization problem). The technique can be used to quickly and reliably obtain initial results, test out a putative fitness function formulation, and to assess the structure of the search landscape. In SBSE, where many new application areas are still being discovered, HC denotes a useful tool: providing fast, reliable, and understandable initial results. It should be tried before more sophisticated algorithms are deployed.

Table III shows the venues in which SBSE publications have appeared. In total the papers on SBSE have appeared in 201 different publication venues, which partly explains why a survey like the present one is needed. This spread of publication venues reveals that there are a wide range of publication outlets for SBSE work. The data also indicate that SBSE work is achieving acceptance in the leading SE journals and conferences as well as those from the SBO and OR communities. This reflects a healthy profile of publication reaching all of the communities to which this work is relevant. It bodes well for the development of the discipline.

Figure 4 presents a formal concept lattice of the literature on SBSE. Formal Concept Analysis [Snelting 1998] is a technique that can be applied to tabular data that report objects, attributes, and the binary relationships between them. A "concept" is a maximal rectangle in the adjacency matrix of objects and attributes, that is, a concept denotes a maximal set of objects that possess a given (also maximal) set of attributes.

The results of FCA are typically displayed as a concept lattice such as that presented in Figure 4. The lattice exploits symmetry properties enjoyed by all concept spaces (the details of which are beyond the scope of this article). These properties have been shown to hold, irrespective of the particular choice of objects and attributes, thereby imbuing FCA with an enduring appeal. In the case of Figure 4, the objectives are application areas and the attributes are the search-based optimization techniques that have been

Table III. Publication Venues

	Network	Requirements	Design	Coding	Verification	Maintenance	Metrics	Management	Agents	Security	General	Testing	Total
GECCO		2	4	1	5	16	2	8	2	1		46	87
TR			3								2	15	20
IST	1	2	1	1				5			2	5	17
PhD Thesis			1			3						11	15
CEC			1		3						2	8	14
ASE												14	14
SBST												12	12
MSc Thesis			1								1	8	10
ICSM		1	1			3		2				2	9
TSE						1		1				6	8
ISSTA												8	8
COMPSAC						1		2			1	3	7
COR								2				5	7
CSMR			1			5							6
SEKE						1		1			2	2	6
JSS						2						4	6
STVR											1	5	6
Book						2						3	5
ICSE											2	3	5
Others	4	6	35	3	13	18	3	22	1	1	4	125	235

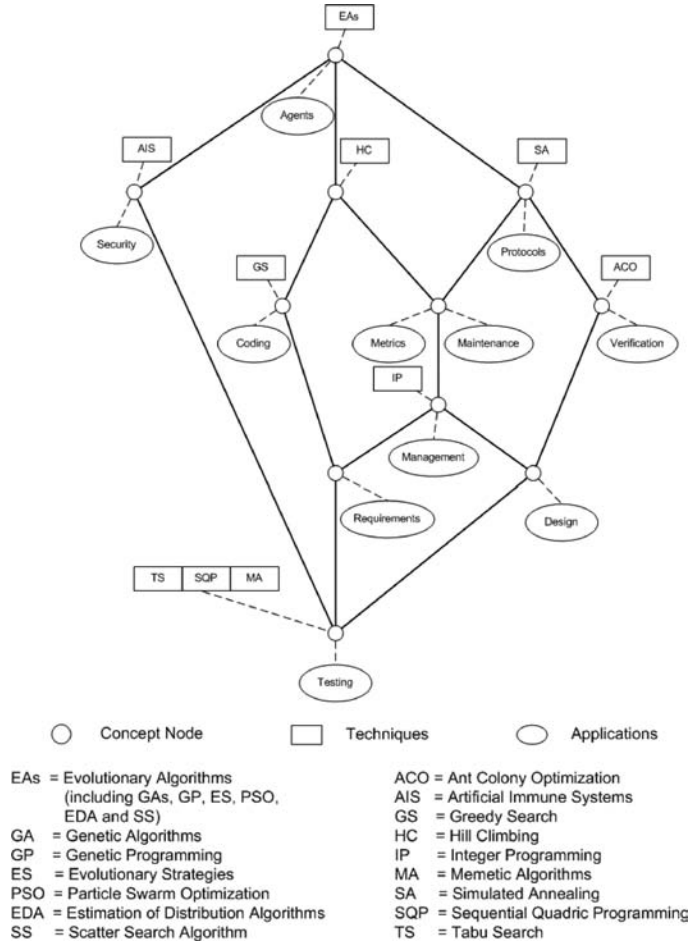


Fig. 4. FCA for techniques and applications in the SBSE literature 1976–2008.

applied to the corresponding application areas. A concept is thus a set of Software Engineering application areas to which a set of search-based optimization techniques have been applied, such that no larger set of areas can be found for the same set of techniques and no larger set of techniques can be found for the same areas.

The names of the SE application areas presented in Figure 4 are abbreviated due to space limitations. That is, "Protocols" is the abbreviation of "Network Protocols (C.2.2)"; "Requirements" is the abbreviation of "Requirements/Specifications (D.2.1)"; "Design" is the abbreviation of "Design Tools and Techniques (D.2.2)"; "Coding" is the abbreviation of "Coding Tools and Techniques (D.2.3)"; "Verification" is the abbreviation of "Software/Program Verification (D.2.4)"; "Testing" is the abbreviation of "Testing and Debugging (D.2.5)"; "Maintenance" is the abbreviation of "Distribution; Maintenance and Enhancement (D.2.7)"; "Agents" is the abbreviation of "Distributed Artificial Intelligence (I.2.11)"; "Security" is the abbreviation of "Security and Protection (K.6.5)".

In the lattice, a concept is denoted by a node. The concepts are related to one another by edges. If a node n_1 is related to a node n_2 (with n_2 higher up the diagram) then this means, in the case of the SBSE lattice of Figure 4, that all the application areas present

at concept n_1 are also present at concept n_2 and that all the optimization techniques present at n_2 are also present at n_1.

It turns out that, for all lattices, there is a unique labeling of nodes, such that an objective and attribute need appear only once in the labeling. In the case of the SBSE lattice, the labels correspond to application areas in SE and optimization techniques. An application area appearing at node n, also implicitly appears at all the nodes reachable from n moving up the lattice. By symmetric counterpart, an application area that appears at a node m in the lattice also implicitly appears at all the nodes reachable from m, traveling down the lattice. Figure 4 includes concepts relating to testing. When these are removed and the lattice recomputed the effect is merely the disappearance of those node labels (all of which relate only to testing) at the lowest node in the lattice. This is merely a reflection of the fact that testing has been optimized using a superset of all optimization techniques applied to other SE problem domains.

The lattice for the SBSE literature reveals a few interesting properties of the clustering of application areas and techniques. First, it is clear that the testing application area has had every optimization technique applied to it in the SBSE literature (because it appears at the bottom of the lattice), while no technique has been applied to every area (indicating that there are still gaps here). Furthermore, four techniques, TS, SQP, MA, and EDA, have *only* been applied so far in Software Testing. Of those techniques so far explored these are the least widely applied.

It is also clear that the most widely applied techniques are SA and EAs, backing up the findings of Figure 3. Hill climbing, though popular, has only been applied to design, maintenance, management, and testing. Only EAs have been applied to agents, while protocols form an interesting link between PSO and SA. They are the only application areas (apart from the ubiquitous area of testing) to which both PSO and SA have been applied.

Figure 4 can also be read like a subsumption diagram. For example, all areas to which IP, HC, and ACO have been applied have also had SA applied to them and all these have had EAs applied to them. Reading the relationship in the other direction, all techniques applied to agents have also been applied to Coding and all these have been applied to Requirements. Readers may also find other relationships in the lattice that are of interest, depending upon the particular areas and techniques that are of interest to them.

10. HOW SBSE REUNITES PREVIOUSLY DIVERGENT AREAS OF SE

In the early development of the field of SE the nascent field split into different topic areas, with many different disjoint communities focusing on different aspects of the emerging discipline. Of course, this has been a natural and necessary evolution of the subject and it was to be expected. However, it has had the disadvantage that it has created silos of activity with few connections between them.

Fortunately, SBSE acts as a catalyst to remove barriers between subareas, thereby combating the disadvantages of "silo mentality". It is interesting to observe how SBSE creates these linkages and relationships between areas in SE that would otherwise appear to be completely unrelated. For instance, the problems of requirements engineering and regression testing would appear to be entirely unrelated topics.

Indeed, these two areas of SE soon developed their own series of conferences, with work on requirements engineering tending to appear in the conference and journal of the same name, while work on regression testing would tend to appear at conferences such as the ACM International Symposium on Software Testing and Analysis and the IEEE International Conference on Software Testing.

However, using SBSE, a clear relationship can be seen between these two problem domains. As optimization problems they are remarkably similar, although they occur

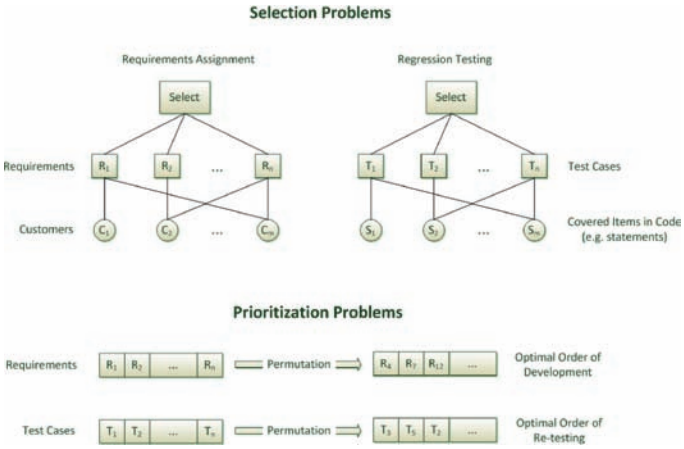


Fig. 5. Requirements selection and regression testing.

at different phases of the software development process and, typically, researchers working within each topic will form disjoint communities.

Figure 5 illustrates the SBSE-inspired relationship between requirements optimization and regression testing. As a selection problem, the task of selecting requirements is closely related to the problem of selecting test cases for regression testing. The difference is that test cases have to cover code in order to achieve high fitness, whereas requirements have to cover customer expectations. In the detail, there will be differences in these two forms of coverage, but as optimization problems, the similarity is striking: both can be viewed as subset selection problems and also as set cover problems.

When one turns to the problem of prioritization, the similarity is also most striking. Both regression test cases and requirements need to be prioritized. In requirement analysis, we seek an order that will ensure that, should development be interrupted, then maximum benefit will have been achieved for the customer at the least cost to the developer, a classic multiobjective cost/benefit problem. For test cases, the prioritization must seek to ensure that, should testing be stopped, then maximum achievement of test objectives is achieved with minimum test effort.

This is an appealing aspect of SBSE. It has the potential to create links and bridges between areas of SE that have grown apart over the years, but which submit to similar analysis from the optimization point of view. Such relationships may lead to exciting new opportunities for cross fertilization between disjoint research communities. These opportunities are a compelling reason for the emergence of conferences and events that focus on Search-Based SE. The approach clearly has the potential to cut across traditional SE boundaries.

11. OVERLOOKED AND EMERGING AREAS

Some areas of SBSE activity have been considered briefly in the literature and then appear to have been overlooked by subsequent research. This section highlights these areas. That is, these are topics that have been addressed, shown promising results, but which have attracted neither follow-on studies nor (relatively speaking) many citations. Given the initially patchy nature of work on SBSE and the recent upsurge in interest and activity, these potentially overlooked areas may be worthy of further study.

Furthermore, this survey comes at a time when SBSE research is becoming widespread, but before it has become mainstream. It is too soon to know whether some of the areas that have apparently hitherto been overlooked might not simply be

emerging areas in which there will be intense activity over the next few years. This section considers both emergent and overlooked areas together; these areas denote either SE subareas or optimization potentialities that remain to be more fully explored.

11.1. Information-Theoretic Fitness

Lutz [2001] considered the problem of hierarchical decomposition of software. The fitness function used by Lutz is based upon an information-theoretic formulation inspired by Shannon [1948]. The function awards high fitness scores to hierarchies that can be expressed most simply (in information-theoretic terms), with the aim of rewarding the more "understandable" designs. The paper by Lutz is one of the few to use information-theoretic measurement as a fitness mechanism. This novel and innovative approach to fitness may have wider SBSE applications.

More recently, Feldt et al. [2008] also used an information-theoretic model, drawing on the observation that the information content of an object can be assessed by the degree to which it can be compressed (this is the so-called Kolmogorov complexity). This recent work may be an indication that information-theoretic fitness is not likely to remain an "overlooked area" for much longer. The authors believe that there is tremendous potential in the use of information theory as a source of valuable fitness for SE; after all, SE is an information-rich discipline, so an information-theoretic fitness function would seem to be a natural choice.

11.2. Optimization of Source Code Analysis

Only a few papers appear to concern source-code-based SBSE. This is likely to be a growth area, since many source-code analysis and manipulation problems are either inherently undecidable or present scalability issues. The source code analysis community has long been concerned with a very rigid model of analysis, in which conservative approximation is the favored approach to coping with the underlying undecidability of the analysis problem.

However, more recently, Ernst's seminal work on the detection of likely invariants [Ernst 2000], which spawned the widely used and influential Daikon tool [Ernst et al. 2001] demonstrated that unsound analyses can yield extremely valuable results. The full potential of this observation has yet to be realized. Through the application of SBSE, it will be possible to search for interesting features and to provide probabalistic source code analyses that, like the Daikon work, may not be sound, but would nonetheless turn out to be useful.

A summary of the papers addressing problems related to Coding Tools and Techniques (ACM: D.2.3) is given in Table VII. All of these papers could be regarded as representing an emerging area of optimization for source-code analysis using SBSE. Hart and Shepperd [2002] addressed the automatic evolution of controller programs by applying GAs to improve the quality of the output vector, while Di Penta et al. [Di Penta et al. 2008; Di Penta and Taneja 2005] proposed a GA-based approach for grammar inference from program examples toward suitable grammars. The grammar captures the subset of the programming language used by the programmer and can be used to understand and reason about programming language idioms and styles.

Jiang et al. [2007b, 2008] used search-based algorithms to decompose the program into slices and to search for useful dependence structures. The search problem involves the space of subsets of program slices, seeking those that denote decomposable but disparate elements of code using metaheuristic search and also greedy algorithms. The results showed that, as procedures become larger, there was a statistically significant trend for them to become also increasingly splittable.

More recently, Zeller [2011] argued for an iterative cycle of program analysis and search based test data generation.

11.3. SBSE for Software Agents and Distributed Artificial Intelligence

Software agents and the general areas known by the term "Distributed Artificial Intelligence" in the ACM classification system would seem to provide a rich source of problems for SBSE, particularly those approaches that use population-based optimization. A summary of the papers addressing Distributed Artificial Intelligence (ACM: I.2.11) is given in Table XII. As can be seen, there is comparatively little work in this area, despite there being some early work by Sinclair and Shami [1997], who investigated the relative efficiency of GAs and GP to evolve a grid-based food gathering agent. More recently, Haas et al. [2005] used a GA for parameter tuning of multiagent systems, while Hodjat et al. [2004] applied GAs to improve agent-oriented natural language interpreters.

This apparent lack of other previous work is something of a surprise since the nature of multiagent systems seems very closely aligned and amenable to SBSE. That is, an agent-based system consists of a population of individuals that interact and share information, seeking to solve a common goal. A population-based optimization algorithm also consists of a set of individuals that exchange information through cross over. Furthermore, coevolutionary optimization seems particularly well suited to the agent-oriented paradigm; each agent could coevolve its beliefs, desires, and intention in coevolutionary cooperation with others. Alternatively, using competitive coevolution, it may be possible to identify good agent designs by creating an environment in which they are subjected to evolutionary pressure, using GP to evolve their internal structure.

The authors believe that the potential for SBSE applications in the area of software agents is enormous. Recent work[3] [Nguyen et al. 2009] demonstrated how an agent can be tested using SBSE techniques. We hope to further develop this model of evolutionary agents.

11.4. Security and Protection

There have been very few papers on the application of SBSE to problems of security. A summary of the papers addressing Security and Protection areas (ACM: K.6.5) is given in Table XIII. This is sure to change, given the importance of this area of application. The challenge is often to find a way to encode a security problem as a fitness function.

Often security aspects have a decidedly boolean character to them; either a security problem is present or it is absent. In order to fully apply SBSE techniques to find security problems, it will be necessary to find a way to formulate fitness functions that offer a guiding gradient toward an optimum.

Some authors have managed to do this. Dozier et al. [2004] described how the design of AIS-based Intrusion Detection Systems (IDSs) can be improved through the use of evolutionary hackers in the form of GENERTIA red teams (GRTs) to discover holes found in the immune system. Dozier et al. [2007] compared a hacker with 12 evolutionary hackers based on PSO that have been used as vulnerability analyzers for AIS-based IDSs. Del Grosso et al. [2005, 2008] showed how SBSE can be used to detect buffer overflow vulnerabilities, thereby helping to guard against "stack smash" attacks.

11.5. Protocols

Protocol correctness, efficiency, security, and cost are all aspects of protocol definitions that can and have been explored using SBSE. Alba and Troya [1996] presented a first attempt in applying a GA for checking the correctness of communication protocols (expressed as a pair of communicating FSMs). Clark and Jacob [2000] used GAs in the

[3]The work is not included in the tables and analysis in this survey since it is published after the census date.

design and development of Burrows, Abadi, and Needham (BAN) protocols optimizing for the trade-off between protocol security, efficiency, and cost. This was subsequently extended by Clark and Jacob [2001], who applied GAs and SA approaches to the problem addressed in Clark and Jacob [2000]. El-Fakih et al. [1999] used the 0-1 ILP and GAs to solve the message exchange optimization problem for distributed applications in order to reduce the communication cost. Ferreira et al. [2008] proposed PSO to detect network protocol errors in concurrent systems. A summary of the papers addressing problems in the area of Network Protocols (ACM: C.2.2) using the search-based approach is given in Table IV.

11.6. Interactive Optimization

All of the fitness functions so far considered in the literature on SBSE have been fully automated. This seems to be a prerequisite; fast fitness computation is needed for repeated evaluation during the progress of the search. However, outside the SBSE domain of application, there has been extensive work on fitness functions that incorporate human judgement [Funes et al. 2004]. This form of search is known as interactive optimization and it is clearly relevant in many aspects of SE, such as capturing inherently intuitive value judgements about design preferences [Simons and Parmee 2008b].

In SE, interactive optimization could be used in a number of ways. Many problems may naturally benefit from human evaluation of fitness. For example, in design problems, the constraints that govern the design process may be ill-defined or subjective. It may also be possible to use a search-based approach to explore the implicit assumptions in human assessment of solutions. For example, by identifying the building blocks that make up a good solution according to a human fitness evaluation, it may be possible to capture otherwise implicit design constraints and desirable features.

The key problem with any interactive approach to optimization lies in the requirement to repeatedly revert to the human for an assessment of fitness, thereby giving rise to possible fatigue and learning-effect bias. If this fatigue problem can be overcome in the SE domain (as it has in other application domains) then interactive optimization offers great potential benefits to SBSE.

Harman [2007a] provided an overview of SBSE for problems in program comprehension, which includes ways in which interactive evolution might be applied in problems relating to code understanding.

11.7. Online Optimization

All applications of SBSE of which the authors are aware concern what might be termed "static" or "offline" optimization problems, that is, problems where the algorithm is executed offline in order to find a solution to the problem in hand. This is to be contrasted with "dynamic" or "online" SBSE, in which the solutions are repeatedly generated in real time and applied during the lifetime of the execution of the system to which the solution applies.

The static nature of the search problems studied in the existing literature on SBSE has tended to delimit the choice of algorithms and the methodology within which the use of search is applied. PSO [Zhang et al. 2005] and ACO [Dorigo and Blum 2005] techniques have not been widely used in the SBSE literature. These techniques work well in situations where the problem is rapidly changing and the current best solution must be continually adapted.

It seems likely that the ever-changing and dynamic nature of many SE problems would suggest possible application areas for ACO and PSO techniques. It is surprising that highly adaptive search techniques like ACO have yet to be applied widely in SBSE. Perhaps distributed, service-oriented, and agent-oriented SE paradigms will provide additional candidate application areas for ACO and PSO.

11.8. SBSE for Nonfunctional Properties

There has been much work on stress testing [Alander et al. 1997b; Briand et al. 2005, 2006; Garousi 2006, 2008; Garousi et al. 2008; Mantere 2003] and temporal testing [Alander et al. 1997a, 1998, 1997b, 1996; Dillon 2005; Groß 2000, 2001; Groß et al. 2000; Groß and Mayer 2002, 2003; Pohlheim and Wegener 1999; Tlili et al. 2006; Wegener and Grochtmann 1998; Wegener et al. 1997a, 1997b; Wegener and Mueller 2001], but far less on other non functional properties such as heat dissipation and power consumption [Joshi et al. 2008; White et al. 2008] and thermal properties such as temperature and heat dissipation [Joshi et al. 2008]. The problem of QoS introduced by Canfora et al. [2005a] also denotes an area of nonfunctional optimization in SE which has recently witnessed an upsurge in activity and interest [Jaeger and Mühl 2007; Ma and Zhang 2008; Su et al. 2007; Zhang et al. 2006, 2007b].

It seems likely that the drive to ever-smaller devices and to massively networked devices will make these issues far more pressing in the future, thereby engendering more research in this area. These are important emergent SE paradigms, though perhaps not widely regarded as current mainstream SE. Afzal et al. [2009] provided a detailed in-depth survey of approaches to testing nonfunctional requirements, to which the reader is referred for a more detailed treatment of this area.

11.9. Multiobjective Optimization

SE problems are typically multiobjective problems. The objectives that have to be met are often competing and contradictory. For example, in project planning, seeking earliest completion time at the least expensive overall cost will lead to a conflict of objectives. However, there is no necessary simple trade-off between the two, making it desirable to find "sweet spots" that optimize both.

Suppose a problem is to be solved that has n fitness function, $f_1, \ldots, f_n$ that take some vector of parameters $\overline{x}$. One simple-minded way to optimize these multiple objectives is to combine them into a single aggregated fitness, F, according to a set of coefficients, $c_i, \ldots, c_n$: $F = \sum_{i=1}^{n} c_i f_i(\overline{x})$. This approach works when the values of the coefficients determine precisely how much each element of fitness matters. For example, if two fitness functions, f_1 and f_2 are combined using $F = 2 \cdot f_1(\overline{x}) + f_2(\overline{x})$ then the coefficients $c_1 = 2, c_2 = 1$ explicitly capture the belief that the property denoted by fitness function f_1 is twice as important as that denoted by fitness function f_2. The consequence is that the search may be justified in rejecting a solution that produces a marked improvement in f_2, if it also produces a smaller reduction in the value of f_1.

Most work on SBSE uses software metrics in one form or another as fitness functions [Harman and Clark 2004]. However, the metrics used are often those that are measured on an *ordinal scale* [Shepperd 1995]. As such, it is not sensible to combine these metrics into an aggregate fitness in the manner described earlier. The use of Pareto optimality is an alternative to aggregated fitness. It is superior in many ways. Under Pareto optimality, one solution is better than (i.e., *dominates*) another if it is better according to at least one of the individual fitness functions and no worse according to all of the others.

When searching for solutions to a problem using Pareto optimality, the search yields a set of solutions that are nondominated, that is, each member of the nondominated set is no worse than any of the others in the set, but also cannot be said to be better. Any set of nondominated solutions forms a Pareto front. Consider Figure 6, which depicts the computation of Pareto optimality for two imaginary fitness functions (objective 1 and objective 2). In the figure, points $S1$, $S2$, and $S3$ lie on the Pareto front, while $S4$ and $S5$ are dominated. Interested readers may refer to Collette and Siarry [2004] for further details about multiobjective optimization and Pareto optimality.

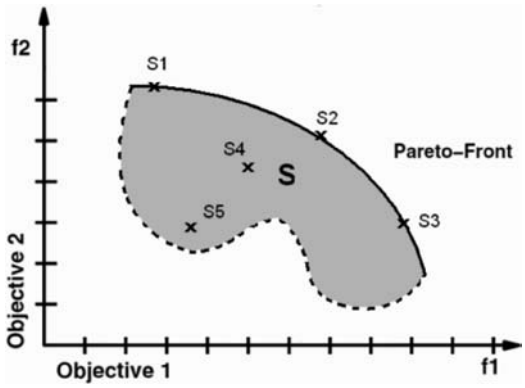


Fig. 6. Pareto optimality and pareto fronts.

Recently, research on SBSE has started to move from single-objective formulations to multiobjective formulations, with an increasing focus on Pareto-optimal optimization techniques. For example, Harman [Yoo and Harman 2011] recently set out a research agenda for Multiobjective Regression Test Optimization. Recent work has produced multiobjective formulations of problems in many application areas within SE including requirements [Finkelstein et al. 2008; Zhang et al. 2007b], testing [Del Grosso et al. 2005; Everson and Fieldsend 2006; Harman et al. 2007b], quality assurance [Khoshgoftaar et al. 2004b], refactoring [Harman and Tratt 2007], and project management [Alba and Chicano 2007d].

11.10. Coevolution

In coevolutionary computation, two or more populations of solutions evolve simultaneously with the fitness of each depending upon the current population of the other. The idea, as so far applied in SBSE work, is to capture a predator-prey model of evolution, in which both evolving populations are stimulated to evolve to better solutions.

Mantere [2003] also proposed a coevolutionary approach to automatically generate test images for the image processing software. Adamopoulos et al. [2004] suggested the application of coevolution in mutation testing, arguing that this could be used to evolve sets of mutants and sets of test cases, where the test cases act as predators and the mutants as their prey. Arcuri et al. [Arcuri 2008; Arcuri and Yao 2007] used coevolution to evolve programs and their test data from specifications using coevolution.

Arcuri and Yao [Arcuri 2008; Arcuri and Yao 2008] also developed a coevolutionary model of bug fixing, in which one population essentially seeks out patches that are able to pass test cases, while test cases can be produced from an oracle in an attempt to find the shortcomings of a current population of proposed patches. In this way the patch is the prey, while the test cases, once again, act as predators. The approach assumes the existence of a specification to act as the oracle.

Coevolution can also be conducted in a cooperative manner, an approach not explored in SBSE until very recently [Ren et al. 2011]. It is likely to be productive in finding ways in which aspects of a system can be coevolved to work better together and, like the previously studied competitive coevolutionary paradigm, offers great potential for further application in SBSE.

Many aspects of SE problems lend themselves to a coevolutionary model of optimization because software systems are complex and rich in potential population that could be productively coevolved (using both competitive and cooperative coevolution). As with traditional SBSE, it is in the area of testing where the analogy is perhaps clearest

and most easily applied, which may be why this area has already been considered in the literature.

Though all of these may not occur in the same systems, they are all the subject of change and, should a suitable fitness function be found, can therefore be evolved. Where two such populations are evolved in isolation, but participate in the same overall software system, it would seem a logical "next step", to seek to evolve these populations together; the fitness of one is likely to have an impact on the fitness of another, so evolution in isolation may not be capable of locating the best solutions. Like the move from single to multiple objectives, the migration from evolution to coevolution offers the chance to bring together theory and real-world reality.

12. FUTURE BENEFITS TO BE EXPECTED FROM OPTIMIZATION IN SE

This section briefly reviews some of the benefits that can be expected to accrue from further development of the field of search-based SE. These benefits are pervading, though often implicit, themes in SBSE research. To borrow the nomenclature of aspect-oriented software development, these are the "cross cutting concerns" of the SBSE world, advantages that can be derived from almost all applications at various points in their use.

12.1. Generality and Applicability

One of the striking features of the SBSE research program that emerges from this survey is the wide variety of different SE problems to which SBSE has been applied. Clearly, testing remains a predominant application, with 54% of all SBSE papers targeting various aspects of testing. However, as the survey reveals, there are few areas of SE activity to which SBO remains unapplied.

This generality and applicability arises from the very nature of SE. The two primary tasks that have to be undertaken before a search-based approach can be applied to an SE problem are the definition of a *representation* of the problem and the *fitness function* that captures the objective or objectives to be optimized. Once these two tasks are accomplished, it is possible to begin to get results from the application of many SBO techniques.

In other engineering disciplines, it may not be easy to represent a problem; the physical properties of the engineering artifact may mean that simulation is the only economical option. This puts the optimization algorithm at one stage removed from the engineering problem at hand. Furthermore, for other engineering disciplines, it may not be obvious how to measure the properties of the engineering artifact to be optimized. Even where the measurements required may be obvious, it may not be easy to collect the readings; once again the physical properties of the engineering materials may be a barrier to the application of optimization techniques.

However, software has no physical manifestation. Therefore, there are fewer problems with the representation of a software artifact, since almost all software artifacts are, by their very nature, based on intangible "materials" such as information, processes, and logic. This intangibility has made many problems for SE. However, by contrast, within the realm of SBSE, it is a significant advantage. There are few SE problems for which there will be no representation, and the readily available representations are often ready to use "out of the box" for SBSE.

Furthermore, measurement is highly prevalent in Software Engineering, with a whole field of research in software metrics that has spawned many conferences and journals. Therefore, it is also unlikely that the would-be search-based software engineer will find him or herself bereft of any putative fitness function.

For these reasons, it is probable that there will be a rapid growth in the breadth of SBSE research. The growth trend revealed by Figure 1(a) is very likely to continue and authors will continue to find ways to bring new SE subareas within the remit of SBSE.

12.2. Scalability

One of the biggest problems facing software engineers is that of scalability of results. Many approaches that are elegant in the laboratory turn out to be inapplicable in the field, because they lack scalability. Fortunately, one of the attractions of the search-based model of optimization is that it is naturally parallelizable. HC can be performed in parallel, with each climb starting at a different point [Mahdavi et al. 2003b]. GAs, being population based, are also naturally parallel; the fitness of each individual can be computed in parallel, with minimal overheads [Asadi et al. 2010; Mitchell et al. 2001]. Search algorithms in general and SBSE in particular, therefore offer a "killer application" for the emergent paradigm of ubiquitous user-level parallel computing.

This trend toward greater parallelism, the need for scalable SE, and the natural parallelism of many SBSE techniques all point to a likely significant development of parallel SBSE to address the issue of SE scalability. Recent work by Yoo et al. [2011b] has also suggested possibilities in the use of General-Purpose Graphics Processing Units (GPGPU) for inexpensive and effective scalability of SBSE problems.

12.3. Robustness

In some SE applications, solution robustness may be as important as solution functionality. For example, it may be better to locate an area of the search space that is rich in fit solutions, rather than identifying an even fitter solution that is surrounded by a set of far less fit solutions.

In this way, the search seeks stable and fruitful areas of the landscape, such that near neighbors of the proposed solution are also highly fit according to the fitness function. This would have advantages where the solution needs to be not merely good enough but also strong enough to withstand small changes in problem character [Beyer and Sendhoff 2007].

Hitherto, research on SBSE has tended to focus on the production of the fittest possible results. However, many application areas require solutions in a search space that may be subject to change. This makes robustness a natural property to which the research community could and should turn its attention.

12.4. Feedback and Insight

False intuition is often the cause of major error in software engineering, leading to misunderstood specifications, poor communication of requirements, and implicit assumptions in designs. SBSE can address this problem. Unlike human-based search, automated search techniques carry with them no bias. They automatically scour the search space for the solutions that best fit the (stated) human assumptions in the fitness function.

This is one of the central strengths of the search-based approach. It has been widely observed that search techniques are good at producing unexpected answers. For example, EAs have led to patented designs for digital filters [Schnier et al. 2004] and the discovery of patented antenna designs [Linden 2002]. Automated search techniques will effectively work in tandem with the human, in an iterative process of refinement, leading to better fitness functions and thereby better encapsulation of human assumptions and intuition.

13. SUMMARY

This article has provided a detailed survey and review of the area of SE activity that has come to be known as SBSE. As the survey shows, the past five years have witnessed a particularly dramatic increase in SBSE activity, with many new applications being addressed.

The article has identified trends in SBSE research, providing data to highlight the growth in papers and the predominance of software testing research. It also indicates that other areas of activities are starting to receive significant attention: requirements, project management, design, maintenance, and reverse engineering predominating. The work also provides a detailed categorization of papers, tabulating the techniques used, the problems studied, and the results presented in the literature to date. This detailed analysis has allowed us to identify some missing areas of activity, some potential techniques that have yet to be applied, and emerging areas.

The future of SBSE is a bright one. There are many areas to which the techniques associated with SBSE surely apply, but have yet to be fully considered. In existing areas of application the results are already very encouraging. Developments emanating from the optimization community will present exciting possibilities, while new challenges from the application domains will present interesting new challenges. If we are to regard software engineering to be truly an *engineering* discipline, then surely we should accept SBSE as a natural consequence; is not *optimization* the cornerstone of all engineering?

A. APPENDIX

Table IV. Papers Addressing Activities Related to Network Protocols

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Alba and Troya [1996]	[1996]	Checking the correctness of communication protocols	Detect deadlock and useless states or transitions	String	GA	Synthetic & Real	Conference: PPSN '96
El-Fakih et al. [1999]	[1999]	Deriving protocols for distributed applications	Minimize communication cost (minimize number of messages to be exchanged)	Vector	0-1 ILP, GA	Synthetic	Conference: PDCS '99
Clark and Jacob [2000]	[2000]	Protocol synthesis	Optimize correctness, cost and efficiency	String	GA	Synthetic	Symposium: S&P '00
Clark and Jacob [2001]	[2001]	Synthesis of secure protocols	Optimize trade-off between security, efficiency and cost	Integer array (SA), bit string (GA)	SA, GA	Synthetic	Journal: Information and Software Technology
Ferreira et al. [2008]	[2008]	Detecting protocol errors	Detect deadlock violations	Graph	PSO	Synthetic	Conference: HPCS '08

Table V. Papers Addressing Activities Related to Requirements/Specifications

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Bagnall et al. [2001]	[2001]	Requirements selection and optimization	Maximize customers' satisfaction	Bit string	GS, HC, SA	Synthetic	Journal: Information and Software Technology
Feather and Menzies [2002]	[2002]	Requirements selection and optimization	Maximize benefit, minimize cost	unknown	SA	Real	Conference: RE '02
Feather et al. [2004]	[2004]	System design optimization	Maximize benefit, minimize cost	Bit string	SA	Real	Symposium: INCOSE '04
Greer and Ruhe [2004]	[2004]	Requirements selection and optimization	Minimize total penalty and maximize total benefit (weighted sum of the two)	Bit string	GA	Real	Journal: Information and Software Technology
Baker et al. [2006]	[2006]	Requirements selection and optimization	Maximize value	Bit string	SA, GS	Real	Conference: ICSM '06
Feather et al. [2006]	[2006]	Visualization techniques to present Requirements status	-	-	SA	Real	Workshop: REV '06
Harman et al. [2006]	[2006]	Feature subset selection	Maximize total value	Bit string	GS	Real	Conference: GECCO '06
Zhang et al. [2007b]	[2007b]	Requirement satisfaction for the NRP	Maximize value, minimize cost	Bit string	GA	Synthetic	Conference: GECCO '07
Finkelstein et al. [2008]	[2008]	Fairness analysis in requirements assignments	Maximize each stakeholder's possible satisfaction	Bit string	MOOA (NSGA-II)	Synthetic & Real	Conference: RE '08
Jalali et al. [2008]	[2008]	Requirements decisions optimization	Maximize the number of attainable requirements, minimize cost	Bit string	GS	Real	Workshop: PROMISE '08
Cortellessa et al. [2008a]	[2008a]	Automated selection of COTS components	Minimize the cost while assuring the satisfaction of the requirements	Vector	IP (LINGO based)	Synthetic	Journal: J.UCS
Zhang et al. [2008]	[2008]	Overview of existing work and challenges on search based requirements optimization	-	Bit string	MOOA (NSGA-II)	Synthetic & Real	Conference: REFSQ '08

Table VI. Papers Addressing Activities Related to Design Tools and Techniques

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Feldt [1998a]	[1998]	Software fault tolerance	Develop multiple software variants	Tree	GP	Real	Technical Report
Feldt [1998b]	[1998]	Software fault tolerance	Develop multiple diverse software versions	Tree	GP	Real	Conference: EUROMICRO '98
Feldt [1998c] N	[1998]	Software fault tolerance	Maximize the software program diversity	Tree	GP	Real	Journal: IEE Software
Monnier et al. [1998]	[1998]	Development of scheduling module for a real-time system	Finding a feasible solution within the time constraints	String	GA	Benchmark problems	Conference: EUROMICRO '98
Feldt [1999]	[1999]	Knowledge acquirement in early software development phases	Prioritize requirements and explore design trade-off	Tree	GP	Real	Workshop: SCASE '99
Lutz [2001]	[2001]	Hierarchical architecture decomposition	Minimize complexity	Tree	GA	Synthetic	Journal of Systems Architecture
Feldt [2002]	[2002]	Interactive software development workbench	New feature and knowledge acquirement	Tree	EA (Biomimetic Algorithms)	Real	Technical Report
Antoniol and Di Penta [2003]	[2003]	Library miniaturization	Minimize inter-library dependencies; minimize the number of objects linked by applications	Bit-matrix	GAs	Real (Open Source)	Conference: ICSM '03
O'Keeffe and Ó Cinnéide [2003] N	[2003]	Automated OO design improvement	Minimize rejected, duplicated and unused methods and featureless classes, maximize abstract classes	unknown	SA	unknown	Conference: PPPJ '03
Stephenson et al. [2003a]	[2003a]	Automatic programming, compiler optimization	Minimize code execution time (the fastest code is the fittest)	Tree	GP	Real	Conference: PLDI '03

Continued on next page

Table VI. Papers on Design Tools and Techniques – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Canfora et al. [2004]	[2004]	QoS-aware service composition	Maximize QoS attributes (availability, reliability); minimize cost and response time	Integer array	GA	Synthetic	Conference: ICSOC '04
Khoshgoftaar et al. [2004b]	[2004]	Design and implementation (improvement of software reliability/ quality assurance)	Maximize MOM performance at four cutoff percentiles; Minimize tree size (bloat control fitness function)	Tree	GP, MOOA	Real	Symposium: METRICS '04
Khoshgoftaar et al. [2004a]	[2004]	Design and implementation (improvement of software reliability/ quality assurance)	Maximize MOM performance at four cutoff percentiles; Minimize tree size (bloat control fitness function)	Tree	GP, MOOA	Real	Journal: IEEE TEC
O'Keeffe and Ó Cinnéide [2004]	[2004]	Automated OO design improvement	Minimize rejected, duplicated and unused methods and featureless classes, maximize abstract classes	unknown	SA	Synthetic	Conference: WoDiSEE '04
Canfora et al. [2005a]	[2005]	QoS of composite services	Maximize QoS attributes (availability, reliability); minimize cost and response time	Integer array	GA, IP	Synthetic	Conference: GECCO '05
Canfora et al. [2005b]	[2005]	QoS-aware replanning of composite services	Maximize QoS attributes (availability, reliability); minimize cost and response time	Integer array	GA	Real	Conference: ICWS '05
Cao et al. [2005a]	[2005]	Web service selection	Minimize the overall cost of each execution path	Integer vector	GA	Synthetic	Conference: CIS '05
Cao et al. [2005b]	[2005]	Web service selection	Minimize the overall cost	Integer vector	GA	Synthetic	Workshop: WINE '05

Continued on next page

Table VI. Papers on Design Tools and Techniques – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Lucas and Reynolds [2005]	[2005]	Learning deterministic finite automata	Maximize correctness of classification	Binary String	HC	Synthetic & Real	Journal: IEEE TPAMI
Amoui et al. [2006]	[2006]	OO software architecture design	Optimize metrics and find the best sequence of transformations	unknown	GA	unknown	Journal: ITIC
Aversano et al. [2006]	[2006]	Design of service composition	Maximize recall for outputs; maximize precision	Tree	GP	Real	Journal: CSSE
Sheu and Chuang [2006]	[2006]	Development of scheduling module for a real-time system	Find a feasible solution within the time constraints	String	GA	Synthetic (via simulation)	Journal: IEEE Transactions on Computers
Simons and Parmee [2006]	[2006]	Design comprehension	Maximize cohesion, minimize coupling	Object-based	GA, EP, NSGA-II	Real	Computation: GECCO '06
Yang et al. [2006]	[2006]	Software integration	Minimize software risk	Binary string	GA	Real	Journal: Information and Software Technology
Zhang et al. [2006]	[2006]	Web services selection	Improve QoS	Matrix	GA	Synthetic	Workshop: DEECS '06
Jaeger and Mühl [2007]	[2007]	Web services selection	Improve the QoS	Vector	GA	Synthetic	Workshop: SOASOC '07
Poulding et al. [2007]	[2007]						Symposium: HASE '07
Simons and Parmee [2007]	[2007]	OO conceptual software design	Maximize Cohesiveness of Methods (COM) metric and number of class	Binary string	GA MOOA (NSGA-II)	Real	Journal: Engineering Optimization
Su et al. [2007]	[2007]	Web services selection	Improve QoS	Matrix	GA	Synthetic	Conference: DAIS '07
Zhang et al. [2007a]	[2007]	Web services selection	Improve QoS	Matrix	GA	Synthetic	Journal: Computer Communications
Arcuri et al. [2008]	[2008]	Non functional property optimization	Minimize non functional property and error	Tree	GP, MOOA (SPEA2)	Synthetic	Conference: SEAL '08
Barlas and El-Fakih [2008]	[2008]	Distributed system design	Optimize the delivery to multiple clients by multiple servers	String	GA	Synthetic (Simulation)	Journal: MTA

Continued on next page

Table VI. Papers on Design Tools and Techniques – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Bhatia et al. [2008]	[2008]	Reusable software component retrieval	Generate rules for classifying components	Graph	ACO	-	Conference: ISEC '08
Bowman et al. [2008]	[2008]	Class responsibility assignment (OO design)	Maximize cohesion, minimize coupling	Integer string	MOGA, RS, GA, HC	Synthetic	Technical Report
Chardigny et al. [2008a]	[2008]	Architecture extraction for OO systems	Improve the quality and the semantic correctness of the architecture	unknown	unknown	-	Conference: ECSA '08
Chardigny et al. [2008b]	[2008]	Architecture extraction for OO systems	Improve the quality and the semantic correctness of the architecture	unknown	SA	Real	Conference: CSMR '08
Desnos et al. [2008]	[2008]	Automatic component substitution	Optimize software reuse and evolution	Tree	BA, BBA	Synthetic	Journal: JSME
Goldsby and Cheng [2008b]	[2008]	Software behavioral model generation	Identify multiple behavioral models and satisfy functional properties	-	Digital evolution (Avida-based)	Real	Conference: GECCO '08
Goldsby and Cheng [2008a]	[2008]	Software behavioral model generation	Identify multiple behavioral models and satisfy functional properties	-	Digital evolution (Avida-based)	Synthetic	Conference: MoDELS '08
Goldsby et al. [2008]	[2008]	Software behavioral model generation and satisfy functional properties	Identify multiple behavioral models and satisfy functional properties	-	Digital evolution (Avida-based)	Real	Conference: ICAC '08
Ma and Zhang [2008]	[2008]	Web service selection	Improve QoS	Matrix	GA	Synthetic	Journal: Computer Networks
Räihä [2008a]	[2008]	Automated architecture design	Improve efficiency, modifiability and complexity	A collection of supergenes	GA	Synthetic	Master Thesis

Continued on next page

Table VI. Papers on Design Tools and Techniques – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Räihä [2008b]	[2008]	Automated architecture design	Improve efficiency, modifiability and complexity	A collection of supergenes	GA	Synthetic	Lic. Thesis
Räihä et al. [2008a]	[2008a]	Automated architecture design	Improve efficiency, modifiability and complexity	A collection of supergenes	GA	Synthetic	Conference: SEAL '08
Räihä et al. [2008b]	[2008]	Automating CIM-to-PIM model trans-formations	Improve efficiency, modifiability and complexity	A collection of supergenes	unknown	unknown	Journal: NJC
Sharma and Jalote [2008]	[2008]	Deploying software components	Maximize performance	unknown	Heuristics	Synthetic	Symposium: CBSE '08
Simons and Parmee [2008a]	[2008]	Software design supporting	Minimize design coupling, Maximize cohesion of classes	String	MOOA (NSGA-II)	Synthetic	Conference: GECCO '08
Simons and Parmee [2008b]	[2008]	Conceptual software design	Maximize cohesion of classes; minimize coupling between classes	Object-based	MOOA (NSGA-II)	Real	Congress: CEC '08
Vijayalakshmi et al. [2008]	[2008]	Component selection in software development	unknown	unknown	GA	unknown	Journal: IJISCM

Table VII. Papers Addressing Activities Related to Coding Tools and Techniques

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Hart and Shepperd [2002]	[2002]	Automatic evolution of controller programs	Maximize the quality of the output vector generated	String	GA	Real	Conference: GECCO '02
Jiang et al. [2007b]	[2007]	Dependence analysis	Maximize code coverage and minimize the degree of overlap between the slices	String	HC, GA, GS	Real	Journal: Information and Software Technology
Di Penta et al. [2008]	[2008]	Grammar inference	Automatic evolution from grammar fragments to target grammar	String	GA	Real	Journal: Soft Computing
Hoste and Eeckhout [2008]	[2008]	Compiler optimization	Find a Pareto optimal trade-off among metrics (total execution time, compilation time, code size, energy comsumption)	unknown	MOGA (improved SPEA)	Benchmarks	Symposium: CGO '08
Jiang et al. [2008]	[2008]	Automatic support for procedure splitability analysis	Minimize the overlap of slices representing procedure components	Binary Matrix	GS	Real (open source)	Conference: WCRE '08

Table VIII. Papers Addressing Software/Program Verification

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Minohara and Tohma [1995]	[1995]	Model checking software reliability growth models	Minimize the errors	Bit string	GA	Real	Symposium: ISSRE '95
Godefroid [1997]	[1997]	Model checking for state space of concurrent systems	Detect deadlocks and assertion violations	unknown	State-less search algorithm (VeriSoft-based)	Real	Symposium: POPL '97
Alba and Chicano [2007b]	[2007]	Model checking for safety properties	Find optimal errors trails in faulty concurrent system	Graph	ACO	Real	Conference: EUROCAST '07
Alba and Chicano [2007c]	[2007]	Model checking for safety errors	Detect errors within low amount of memory and CPU time	Graph	ACO	Real	Conference: GECCO '07
Alba and Chicano [2007a]	[2007]	Model checking (Refutation of safety properties)	Find deadlock states	Graph	ACO	Real	Conference: GECCO '07
Johnson [2007]	[2007]	Model checking	Maximize the number of program statements that are satisfied	List	ES	Synthetic	Conference: EuroGP '07
Kiper et al. [2007]	[2007]	V&V process for critical systems	Maximize the chances of mission success	unknown	SA, GA	-	Conference: GECCO '07
Afzal and Torkar [2008b]	[2008]	Software reliability growth modeling	Measure the suitability of GP evolved SRGM	Tree	GP	-	Symposium: CSA '08
Afzal and Torkar [2008a]	[2008]	Software reliability growth modeling	Measure the adaptability and predictive accuracy of GP evolved model	Tree	GP	Real	Conference: ICSEA '08
Afzal et al. [2008b]	[2008]	Software reliability growth modeling	Measure the adaptability and predictive accuracy of GP evolved model	Tree	GP	Real	Conference: INMIC '08
Alba et al. [2008]	[2008]	Model checking for finding deadlock in concurrent program	Detecting the shortest paths that lead to deadlocks	Graph	GA	Real	Conference: GECCO '08
Chicano and Alba [2008b]	[2008]	Model checking for liveness property	Discover liveness errors; Minimize the required resources	Graph	ACO	Real	Congress: WCCI '08

Continued on next page

Table VIII. Papers on Software/Program Verification – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Chicano and Alba [2008a]	[2008]	Model checking for safety property	Discover safety property violations	Graph	ACO	Real	Journal: Information Processing Letters
Chicano and Alba [2008c]	[2008]	Model checking for Liveness property	Improve the efficacy and efficiency of searching for liveness property violations	Graph	ACO	Real	Conference: GECCO '08
He et al. [2008]	[2008]	Verification and generation of programs	unknown	String of verified components	GP	Synthetic	Congress: CEC '08
Hsu et al. [2008]	[2008]	Software Reliability Growth Modeling	Maximize LSE and MLE	Floating-point	GA	Real	Symposium: ISSRE '08
Katz and Peled [2008b]	[2008]	Model checking	Automatic generation of concurrent programs to detect the property violations	Tree	GP	Synthetic	Conference: TACAS '08
Katz and Peled [2008a]	[2008]	Model checking	Automatic generation of concurrent programs to detect the property violations	Tree	GP	Synthetic	Symposium: ATVA '08
Shyang et al. [2008]	[2008]	Model checking	Detect the locating of deadlocks	Bit string	GA	Real	Congress: CEC '08
Wang et al. [2008]	[2008]	Testing resource allocation	Maximize software reliability, minimize cost	unknown	MOOA (NSGA-II, MODE)	Synthetic	Congress: CEC '08

Table IX. Papers Addressing Distribution, Maintenance and Enhancement Activities

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Mancoridis et al. [1998]	[1998]	Software structure clustering	Maximize inter-connectivity (high cohesion); minimize intra-connectivity (low coupling)	Array of module identifiers	HC, GA	Synthetic	Workshop: IWPC '98
Nisbet [1998]	[1998]	Compiler optimization via determining program transformation sequence	Minimize execution time	String	GA	Synthetic	Conference: HPCN '98
Williams [1998] N	[1998]	Automatic parallelization	unknown	unknown	EAs	unknown	PhD Thesis
Cooper et al. [1999]	[1999]	Compiler optimization	Find customized compiler optimization sequences; minimize the size of the object code	String	GA	Benchmark programs in FORTRAN and C	Workshop: LCTES '99
Doval et al. [1999]	[1999]	Software module clustering	Maximize inter-connectivity (high cohesion); minimize intra-connectivity (low coupling)	Array of module identifiers	GA	Real	Conference: STEP '99
Mancoridis et al. [1999]	[1999]	Automatic clustering of system structure	Maximize inter-connectivity (high cohesion); minimize intra-connectivity (low coupling)	Array of module identifiers	GA	Synthetic	Conference: ICSM '99
Ryan [2000] N	[2000]	Automatic reengineering of software	unknown	unknown	GP	Real	Book
Harman et al. [2002]	[2002]	Software module clustering	Optimize granularity, cohesion and coupling metrics	Array of module identifiers	GA	Synthetic	Conference: GECCO '02
Mitchell [2002]	[2002]	Software structure clustering	Maximize inter-connectivity (high cohesion); minimize intra-connectivity (low coupling)	Array of module identifiers	HC, GA, Exhaus-tive search	Real	PhD Thesis
Mitchell and Mancoridis [2002]	[2002]	Software module clustering via improved HC	Optimize MQ metric	Array of module identifiers	HC	Real	Conference: GECCO '02

Continued on next page

Table IX. Papers on Distribution, Maintenance and Enhancement Activities – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Mitchell et al. [2002]	[2002]	Reverse engineering from source code	Extract design structure	Array of module identifiers	HC, GA, Exhaustive search	Real (open source)	Conference: SEKE '02
Sahraoui et al. [2002] N	[2002]	Object identification in legacy code	Minimize coupling and maximize cohesion	Array of sets of data representing candidate objects	GA	Real	Conference: COMPSAC '02
Van Belle and Ackley [2002]	[2002]	Code factoring	Performance on a randomly changing symbolic regression problem	Tree	GP	Synthetic	Conference: GECCO '02
Antoniol et al. [2003]	[2003]	Libraries refactoring	Minimize the Dependency Factor (DF), Linking Factor (LF) and Standard Deviation Factor (SF)	Bit-matrix	GA	Real	Conference: CSMR '03
Fatiregun et al. [2003]	[2003]	Program transformation	Find the optimum sequence of transformation rules	Sequence of transformation identifiers	HC	-	Conference: GECCO '03
Mahdavi et al. [2003a]	[2003]	Software module clustering	Maximize the number of internal edges; minimize the number of external edges	Array of module identifiers	HC, GA, SA	Real	Conference: GECCO '03
Mahdavi et al. [2003b]	[2003]	Software module clustering	Maximize the number of internal edges; minimize the number of external edges	Array of module identifiers	HC	Real	Conference: ICSM '03
Mitchell and Mancoridis [2003]	[2003]	Software clustering	Maximize inter-connectivity (high cohesion); minimize intra-connectivity (low coupling)	Array of module identifiers	HC, GA, Exhaustive search	Real	Conference: GECCO '03
Reformat et al. [2003]	[2003]	Automatic generation of clones	Measure feasibility of using GP for clone generation	Parse tree	GP	Real (open source)	Conference: ICTAI '03
Stephenson et al. [2003b]	[2003b]						Conference: EuroGP '03
Cowan et al. [2004] N	[2004]	Evolutionary programming	unknown	unknown	GP	unknown	Book
Fatiregun et al. [2004]	[2004]	Program transformation	Minimize program size	Sequence of transformation identifiers	GA, HC, RS	Synthetic	Workshop: SCAM '04

Continued on next page

Table IX. Papers on Distribution, Maintenance and Enhancement Activities – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Mitchell et al. [2004]	[2004]	Software module clustering using Bunch tool	Maximize inter-connectivity (high cohesion); minimize intra-connectivity (Low coupling)	Array of module identifiers	HC, SA	Synthetic	Conference: GECCO '04
Di Penta [2005]	[2005]	Software system renovation	Trade-off among Dependency Factor (DF), Partitioning Ratio (PR), Standard Deviation Factor (SDF)	Bit-matrix	GA	Real	Conference: CSMR '05
Di Penta et al. [2005]	[2005]	Software system renovation	Trade-off among Dependency Factor (DF), Partitioning Ratio (PR), Standard Deviation Factor (SDF) and Feedback Factor (FF)	Bit-matrix	GA, HC	Real	Journal of Systems and Software
Di Penta and Taneja [2005]	[2005]	Automatic grammar evolution	Optimize the percentage of positive examples and negative examples	Array of symbols	GA	Real	Conference: CSMR '05
Fatiregun et al. [2005]	[2005]	Amorphous slicing	Minimize size of the amorphous slice computed	Sequence of transformation identifiers	GA, HC	Real (6 programs)	Conference: WCRE '05
Harman et al. [2005]	[2005]	Software module clustering	Optimize MQ and EVM (comparison of two fitness)	Array of module identifiers	GA	Synthetic & Real	Conference: GECCO '05
Mahdavi [2005]	[2005]	Comprehension via module clustering	Maximize modularization quality (maximize cohesion and minimize coupling)	Array of module identifiers	GA, HC	Real	PhD Thesis
Seng et al. [2005]	[2005]	System re-structuring	Optimize weighted sum of: coupling, cohesion, complexity, cycles, bottlenecks	String	GA	Real	Conference: GECCO '05

Continued on next page

Table IX. Papers on Distribution, Maintenance and Enhancement Activities – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Sutton et al. [2005]	[2005]	Clone detection	Minimizing number of individuals and Maximizing similarity of clones in each individual	Variable-sized vectors	EA	Real (small case)	Conference: GECCO '05
Bate and Emberson [2006]	[2006]	Allocation and scheduling tasks in real-time embedded systems	Improve the flexibility	String	SA	Synthetic	Symposium: RTAS '06
Bouktif et al. [2006a]	[2006]	Mutation and coverage testing	Clone refactoring	String (binary)	GA, ACO, TS	Real	Conference: GECCO '06
Cohen et al. [2006]	[2006]	Thread clustering	Maximize modularization quality	Array of module identifiers	HC	Benchmark example	Conference: GECCO '06
Del Rosso [2006]	[2006]	Dynamic memory configuration	Improve memory efficiency (find the optimal configuration for the segregated free lists)	String	GA	Simulator	Workshop: WISER '06
Gold et al. [2006]	[2006]	Overlapping concept assignment	Maximize quality of concept binding	String (a set of segment pairs)	GA, HC	Real	Conference: ICSM '06
Harman [2006]	[2006]	SBSE for maintenance and reengineering	-	-	-	-	Conference: CSMR '06
Mitchell and Mancoridis [2006]	[2006]	Software module clustering	Maximize modularization quality (maximize cohesion and minimize coupling)	Array of module identifiers	HC, SA	Real	Journal: IEEE TSE
O'Keeffe and Ó Cinnéide [2006]	[2006]	Refactoring of OO programmes	Maximize design quality metric (three metrics were examined)	unknown	HC, SA	Real	Conference: CSMR '06
Seng et al. [2006]	[2006]	Refactoring	Optimize weighted sum of: coupling, cohesion, complexity, stability	String	EA	Real	Conference: GECCO '06

Continued on next page

Table IX. Papers on Distribution, Maintenance and Enhancement Activities – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Bodhuin et al. [2007a]	[2007a]	Model refactoring	Maximize cohesion and minimize coupling	Bit string	GA	Synthetic	Workshop: WRT '07
Bodhuin et al. [2007b]	[2007b]	Re-packaging download-able applications (Software clustering)	Minimize the packaging size and the average of downloading times	Integer array	GA	Real	Conference: CASCON '07
Dubach et al. [2007]	[2007]						Conference: Computing Frontiers '07
Emberson and Bate [2007]	[2007]	Task allocation and scheduling in mode transitions	Minimize changes in allocation	unknown	SA	Synthetic	Symposium: RTAS '07
Harman and Tratt [2007]	[2007]	Refactoring	Maximize coupling between objects (CBO) and Minimize STDV of the number of methods in classes	Sequence of method moves	HC	Real	Conference: GECCO '07
Huynh and Cai [2007]	[2007]	Software modularity analysis (Clustering)	Check the consistency between design and source code	unknown	GA	Synthetic	Workshop: ACoM '07
O'Keeffe and Ó Cinnéide [2007]	[2007]	Software refactoring	QMOOD hierarchical design quality model	Binary string	HC, SA, GA	Real (Open source)	Conference: GECCO '07
Reformat et al. [2007]	[2007]	Software cloning	Produce a system from its external interactions	Tree	GP	Synthetic	Journal: Soft Computing
Kessentini et al. [2008]	[2008]	Model trans-formation	Maximize completeness and consistency of source model transformation	M-dimensional vector	PSO	Synthetic	Conference: MODELS '08
Kuperberg et al. [2008]	[2008]	Performance prediction	Create platform-independent parametric performance models	Tree	GP	Synthetic	Symposium: CBSE '08

Continued on next page

Table IX. Papers on Distribution, Maintenance and Enhancement Activities – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Mitchell and Mancoridis [2008]	[2008]	Software clustering	Improve modularization quality	Sequence of transformation identifiers	SA	-	Journal: Soft Computing
O'Keeffe and Ó Cinnéide [2008a]	[2008]	Software refactoring	QMOOD hierarchical design quality model	Binary string	SA, GA, HC	Real (Open source)	Journal: JSME
O'Keeffe and Ó Cinnéide [2008b]	[2008]	Software refactoring	QMOOD hierarchical design quality model	Binary string	HC, SA	Real (Open source)	Journal of Systems and Software
White et al. [2008]	[2008]	Non functional properties satisfaction	Trade-offs between power consumption and functionality	Binary	GP, MOOA (SPEA2)	Synthetic	Conference: GECCO '08

Table X. Papers Addressing Activities Related to Metrics

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Harman and Clark [2004]	[2004]	Overview of search based approaches and metrics to SBSE	Guidelines to define good fitness functions	-	HC, SA, GA	-	Symposium: METRICS '04
Vivanco and Pizzi [2004]	[2004]	Software metrics classification	Improve the prediction of software object quality	Bit string	GA	Real	Conference: GECCO '04
Lange and Mancoridis [2007]	[2007]	Metrics classification (Identify software developer based on style metrics)	Maximize the correct classification	String	GA	Real (Open source)	Conference: GECCO '07
Vivanco and Jin [2007]	[2007]	OO source code metrics selection	LDA classifier	Bit-mask	GA	Real	Conference: OOPSLA '07
Vivanco and Jin [2008]	[2008]	Software metrics selection	Improve the prediction of software object quality	Bit string	GA	Real	Symposium: ESEM '08

Table XI. Papers Addressing Management Activities

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Chao et al. [1993] N	[1993]	Software project management	unknown	unknown	GA	unknown	Conference: IAST '93
Chang [1994]	[1994]	Software project management	-	-	GA	-	Journal: IEEE Software
Chang et al. [1994]	[1994]	Software project management	Determine the resource allocation	unknown	GA	-	Conference: COMPSAC '94
Chang et al. [1998]	[1998]	Project scheduling and resource allocation	Minimize total cost and finishing time	String	GA	Synthetic	Conference: COMPSAC '98
Dolado and Fernandez [1998]	[1998]	Software development effort estimation	Optimize effort estimation	Tree	GP, NN	Synthetic & Real	Conference: INSPIRE '98
Evett et al. [1999]	[1999]	Software quality modeling	Predict software quality	Tree	GP	Real	Conference: FLAIRS '99
Dolado [2000]	[2000]	Software size estimation	Optimize software size estimation	Tree	GP, NN	Real? (from literature)	Journal: IEEE TSE
Shukla [2000]	[2000]	Development effort estimation	Maximize precision of effort estimation	String	NN and GA (GA to train NN)	Real	Journal: Information and Software Technology
Aguilar-Ruiz et al. [2001]	[2001]	Project Management	Maximize classification percentage and coverage of the rules	String	EA	Real	Journal: Information and Software Technology
Burgess and Lefley [2001]	[2001]	Software effort estimation	Maximize accuracy of estimation	Tree	GP	Real (from an existing database)	Journal: Information and Software Technology
Chang et al. [2001]	[2001]	Project scheduling and resource allocation	Minimize duration and cost of project, Maximize quality of product	graph	GA	Synthetic	Journal: Annals of Software Engineering
Dolado [2001]	[2001]	Cost estimation	Optimize cost estimation	Tree	GP	Real	Journal: Information and Software Technology

Continued on next page

Table XI. Papers on Management Activities – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Jarillo et al. [2001]	[2001]	Software development effort estimation	Predict the number of defects, estimate the reliability in terms of time and failure	unknown	GA, GP	Real	Conference: OOIS '01
Liu and Khoshgoftaar [2001]	[2001]	Software quality classification model	Minimize the cost of misclassification	Tree	GP	Real	Symposium: HASE '01
Aguilar-Ruiz et al. [2002]	[2002]	Software development effort, time and quality estimation	Maximize accuracy of estimation	Float string	EA	Real? (from literature)	Conference: GECCO '02
Bouktif et al. [2002]	[2002]	Software quality prediction	Improve correctness	Binary tree	GA	Real	Conference: ICSM '02
Kirsopp et al. [2002]	[2002]	Project effort estimation	Optimize effort estimate precision	String	HC, Forward sequential selection	Synthetic	Conference: GECCO '02
Shan et al. [2002]	[2002]	Software development effort estimation	Determine metric sets and improve the prediction of development effort	Tree	GP	Real	Conference: ICCCSWSE '02
Bouktif et al. [2004]	[2004]	Software quality prediction	Improve correctness	Binary tree	GA	Real	Journal of Object Technology
Khoshgoftaar et al. [2003]	[2003]	Software quality classification	Minimize the cost misclassification; minimize the size of decision tree	Tree	GP	Real	Conference: ICTAI '03
Lefley and Shepperd [2003]	[2003]	Software development effort estimation	Maximize accuracy of estimation	Tree	GP	Real	Conference: GECCO '03
Liu and Khoshgoftaar [2003]	[2003]	Software quality classification	Minimize the cost of misclassification; minimize the size of decision tree	Tree	GP	Real	Conference: GECCO '03
Antoniol et al. [2004a]	[2004]	Project planning	Minimize project duration	String	GA, Queuing theory, Simulation	Real	Symposium: METRICS '04

Continued on next page

Table XI. Papers on Management Activities – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Antoniol et al. [2004b]	[2004]	Project planning	Minimize project duration	String (two types)	HC, SA, GA	Real	Conference: GECCO '04
Liu and Khoshgoftaar [2004]	[2004]	Software quality classification	Minimize the cost of misclassification; minimize the size of decision tree	Tree	GP	Real	Symposium: HASE '04
Alba and Chicano [2005]	[2005]	Project management of the whole SE activities	Minimize project duration and cost (conflicting objectives)	Bit string	GA	Synthetic	Conference: MIC '05
Antoniol et al. [2005]	[2005]	Project planning	Minimize project duration	String	GA, HC, SA, Random search	Real	Conference: ICSM '05
Lokan [2005]	[2005]	Software effort estimation	MSE, LAD, MRE, MER and Z,	Symbolic expression	GP	Real	Symposium: METRICS '05
Alvarez-Valdés et al. [2006]	[2006]	Project scheduling	Minimize finish time	String	SSA	Synthetic	Journal of Heuristics
Bouktif et al. [2006b]	[2006]	Quality planning	Maximize the predictive accuracy	String or matrix?	SA	Real	Conference: GECCO '06
Sheta [2006]	[2006]	Development effort estimation	Maximize precision of effort prediction	unknown	GA	Real	Journal of Computer Science
Alba and Chicano [2007d]	[2007]	Project management of the whole SE activities	Minimize project duration and cost (conflicting objectives)	String	GA	Synthetic	Journal: Information Sciences
Khoshgoftaar and Liu [2007]	[2007]	Software quality classification model	Minimize the modified expected cost of misclassification, optimize the number of predicted fault-prone modules and minimize the size of the decision tree model	Tree	GP	Real	Journal: IEEE Transactions on Reliability
Barreto et al. [2008]	[2008]	Staffing software project	Maximize the value creation for project	-	BB	Synthetic	Journal: Computers & Operations Research

Continued on next page

Table XI. Papers on Management Activities – *continued from previous page*

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Chang et al. [2008]	[2008]	Software project scheduling	Minimize the input (overload, project costs, and time)	unknown	GA	Synthetic	Journal: Information and Software Technology
Cortellessa et al. [2008b]	[2008b]	Decision support for software architecture	Minimize cost under delivery time and quality constraints	unknown	IP (LINGO based)	Synthetic	Journal: Computers & Operations Research
Hericko et al. [2008]	[2008]	Team size optimization	Define team size with minimal project effort	-	gradient method	Real	Journal: Information Processing Letters
Huang et al. [2008]	[2008]	Software effort estimation	Minimize the mean magnitude relative error (MMRE)	String	GA	Real	European Journal of Operational Research
Kapur et al. [2008]	[2008]	Staffing for product release	Provide best quality to customers under time constraint	Binary	GA	Synthetic	Journal: JSME
Khoshgoftaar et al. [2008]	[2008]	Software quality classification modeling	Minimize the cost misclassification; minimize the size of decision tree	Integer or real values string	GA	Real	Conference: SEKE '08
Wen and Lin [2008]	[2008]	Multistage human resource allocation	Minimize the project duration; minimize the project cost	Improved fixed-length encoding	GA	Synthetic	Journal: DOAJ

Table XII. Papers Addressing Distributed Artificial Intelligence

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Sinclair and Shami [1997]	[1997]	Design and Implementation (evolution of simple software agents)	Maximize the number of accumulated units	String, Tree	GA, GP	Synthetic	Conference: GALESIA '97
Hodjat et al. [2004]	[2004]	Adaptive agent Oriented Software Architecture (AAOSA)	Improve the success rate and quality of the policies	unknown	GA	Synthetic	Conference: GECCO '04
Haas et al. [2005]	[2005]	Parameter tuning for multi-agent systems	Improve the quality of composition for software configuration	Integer string	GA	Synthetic	Conference: GECCO '05

Table XIII. Papers Addressing Security and Protection

Authors [Ref]	Year	Activity	Objective/ Fitness	Representation method	Search technique	Test problem	Venue
Dozier et al. [2004]	[2004]	Hole discovery in intrusion detection systems (IDS)	Minimize failed detections	String	EA, PSO	Real (Simulated)	Conference: GECCO '04
Dozier et al. [2007]	[2007]	Hole discovery in intrusion detection systems (IDS)	Minimize failed detections	String	EA, PSO	Real (Simulated)	Journal: Applied Soft Computing

Table XIV. Papers Addressing General Aspects of SBSE

Authors [Ref]	Year	Content	Venue
Clark et al. [2000] N	[2000]	Describing several applications of metaheuristic search techniques in SE	Technical Report
Harman and Jones [2001a]	2001	Introduction of SBSE	Journal: Information and Software Technology
Harman and Jones [2001b]	2001	SEMINAL: Software Engineering using Metaheuristic INovative Algorithms	Journal: Information and Software Technology
Harman and Jones [2001c]	2001	Outlining the papers presented at the SEMINAL Workshop and the discussions	Journal: ACM SEN
Pedrycz [2002]	[2002]	Application of Computational Intelligence in different stages of SE	Conference: SEKE '02
Clark [2003]	[2003]	Cryptography using nature-inspired search techniques	Congress: CEC '03
Clark et al. [2003]	[2003]	Reformulating SE as a search problem	Journal: IEE Proceedings - Software
Harman and Wegener [2004]	[2004]	On application of search techniques in SE	Conference: ICSE '04
McMinn [2004]	[2004]	Survey of search based test data generation	Journal: STVR
Rela [2004]	[2004]	A review of EC in all SE activities	Master Thesis
Mantere and Alander [2005]	[2005]	A review of Evolutionary SE	Journal: Applied Soft Computing
Jiang [2006]	[2006]	A review of applying GA to SE problems	Conference: COMPSAC '06
Harman [2007b]	2007	Introducing 8 specific application areas	Conference: ICSE/FOSE '07
Harman [2007a]	2007	A introduction of SBSE for program comprehension	Conference: ICPC '07
Jiang et al. [2007a]	[2007a]	A measure to predict the hardness of GAs to the optimization problem in SE	Congress: CEC '07
Afzal et al. [2008a]	2008	A Review of the articles based on non functional search based software testing in 1996-2007	Conference: SEKE '08
Alander [2008]	[2008]	A bibliography and collection of GA papers applying to testing problems	Technical Report

ACKNOWLEDGMENT

The authors are deeply indebted to numerous colleagues from the growing international SBSE community who have reviewed and commented upon earlier drafts of this article and provided additional details regarding their work and that of others, as well as highlighting corrections and omissions. The authors would also like to thank the anonymous referees for their detailed, thoughtful, and constructive comments on the first draft of this work.

REFERENCES

ADAMOPOULOS, K., HARMAN, M., AND HIERONS, R. M. 2004. How to overcome the equivalent mutant problem and achieve tailored selective mutation using co-evolution. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'04)*. Lecture Notes in Computer Science, vol. 3103. Springer, 1338–1349.

AFZAL, W. AND TORKAR, R. 2008a. A comparative evaluation of using genetic programming for predicting fault count data. In *Proceedings of the 3rd International Conference on Software Engineering Advances (ICSEA'08)*. IEEE, 407–414.

AFZAL, W. AND TORKAR, R. 2008b. Suitability of genetic programming for software reliability growth modeling. In *Proceedings of the International Symposium on Computer Science and its Applications (CSA'08)*. IEEE, 114–117.

AFZAL, W., TORKAR, R., AND FELDT, R. 2008a. A systematic mapping study on non-functional search-based software testing. In *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE'08)*. 488–493.

AFZAL, W., TORKAR, R., AND FELDT, R. 2008b. Prediction of fault count data using genetic programming. In *Proceedings of the 12th IEEE International Multitopic Conference (INMIC'08)*. IEEE, 349–356.

AFZAL, W., TORKAR, R., AND FELDT, R. 2009. A systematic review of search-based testing for non-functional system properties. *Inf. Softw. Technol. 51,* 6, 957–976.

AGUI LAR-RUIZ, J. S., RAMOS, I., RIQUELME, J. C., AND TORO, M. 2001. An evolutionary approach to estimating software development projects. *Inf. Softw. Technol. 43*, 14, 875–882.

AGUILAR- RUIZ , J. S., RI QUELME, J. C., AND RAMOS, I. 2002. Natural evolutionary coding: An application to estimating software development projects. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'02)*. 1–8.

ALANDER, J. T. 2008. An indexed bibliography of genetic algorithms in testing. Tech. rep. 94-1-TEST, University of Vaasa, Vaasa, Finland. May.

ALANDER, J. T., MANTERE, T., AND MOGHADAMPOUR, G. 1997a. Testing software response times using a genetic algorithm. In *Proceedings of the 3rd Nordic Workshop on Genetic Algorithms and their Applications (3NWGA)*. 293–298.

ALANDER, J. T., MANTERE, T., MOGHADAMPOUR, G., AND MATILA, J. 1998. Searching protection relay response time extremes using genetic algorithm-software quality by optimization. *Electr. Power Syst. Res. 46,* 3, 229–233.

ALANDER, J. T., MANTERE, T., AND TURUNEN, P. 1997b. Genetic algorithm based software testing. In *Proceedings of the 3rd International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'97)*. Springer, 325–328.

ALANDER, J. T., MANTERE, T., TURUNEN, P., AND VIROLAINEN, J. 1996. GA in program testing. In *Proceedings of the 2nd Nordic Workshop on Genetic Algorithms and their Applications (2NWGA)*. 205–210.

AL BA, E. AND CHICANO, F. 2005. Management of software projects with GAs. In *Proceedings of the 6th Metaheuristics International Conference (MIC'05)*. Elsevier, 13–18.

ALBA, E. AND CHICANO, F. 2007a. ACOhg: Dealing with huge graphs. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*. ACM Press, New York, 10–17.

ALBA, E. AND CHICANO, F. 2007b. Ant colony optimization for model checking. In *Proceedings of the 11th International Conference on Computer Aided Systems Theory (EUROCAST'07)*. Lecture Notes in Computer Science, vol. 4739. Springer, 523–530.

ALBA, E. AND CHICANO, F. 2007c. Finding safety errors with ACO. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*. ACM Press, New York, 1066–1073.

ALBA, E. AND CHICANO, F. 2007d. Software project management with GAs. *Inf. Sci. 177*, 11, 2380–2401.

ALBA, E., CHICANO, F., FERREIRA, M., AND GOMEZ-PULIDO, J. A. 2008. Finding deadlocks in large concurrent Java programs using genetic algorithms. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08)*. ACM Press, New York, 1735–1742.

ALBA, E. AND TROYA, J. M. 1996. Genetic algorithms for protocol validation. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature (PPSN'96)*. Springer, 870–879.

AL BOURAE, T., RUHE, G., AND MOUSSAVI, M. 2006. Lightweight replanning of software product releases. In *Proceedings of the 1st International Workshop on Software Product Management (IWSPM'06)*. IEEE, 27–34.

ALI, S., BRIAND, L. C., HEMMATI, H., AND PANESAR-WALAWEGE, R. K. 2010. A systematic review of the application and empirical investigation of search-based test-case generation. *IEEE Trans. Softw. Engin.* (to appear).

ALVAREZ-VALDES, R., CRESPO, E., TAMARIT, J. M., AND VILLA, F. 2006. A scatter search algorithm for project scheduling under partially renewable resources. *J. Heurist. 12*, 1-2, 95–113.

Amoui, M., Mirarab, S., Ansari, S., and Lucas, C. 2006. A genetic algorithm approach to design evolution using design pattern transformation. *Int. J. Inf. Technol. Intell. Comput. 1*, 2, 235–244.

Antoniol, G. and Di Penta, M. 2003. Library miniaturization using static and dynamic information. In *Proceedings of the 19th International Conference on Software Maintenance (ICSM'03)*. IEEE, 235–244.

Antoniol, G., Di Penta, M., and Harman, M. 2004a. A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. In *Proceedings of the 10th International Symposium on the Software Metrics (METRICS'04)*. IEEE, 172–183.

Ant Oniol, G., Di Penta, M., and Harman, M. 2004b. Search-Based techniques for optimizing software project resource allocation. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'04)*. Lecture Notes in Computer Science, vol. 3103. Springer, 1425–1426.

Antoniol, G., Di Penta, M., and Harman, M. 2005. Search-Based techniques applied to optimization of project planning for a massive maintenance project. In *Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05)*. IEEE, 240–249.

Antoniol, G., Di Penta, M., and Neteler, M. 2003. Moving to smaller libraries via clustering and genetic algorithms. In *Proceedings of the 7th European Conference on Software Maintenance and Reengineering (CSMR'03)*. IEEE, 307–316.

Arcuri , A. 2008. On The automation of fixing software bugs. In *Proceedings of the Doctoral Symposium of the IEEE International Conference on Software Engineering (ICSE'08)*. ACM Press, New York, 1003–1006.

Arcuri, A., White, D. R., and YAO, X. 2008. Multi-Objective improvement of software using coevolution and smart seeding. In *Proceedings of the 7th International Conference on Simulated Evolution and Learning (SEAL'08)*. Springer, 61–70.

Arcuri, A. and Yao, X. 2007. Coevolving programs and unit tests from their specification. In *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'07)*. ACM Press, New York, 397–400.

Arcuri, A. and Yao, X. 2008. A novel co-evolutionary approach to automatic software bug fixing. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'08)*. IEEE, 162–168.

Asadi, F., Antoniol, G., and Gueheneuc, Y. - G. 2010. Concept locations with genetic algorithms: A comparison of four distributed architectures. In *Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE'10)*. IEEE, 153–162.

Aversano, L., Di Penta, M., and Taneja, K. 2006. A genetic programming approach to support the design of service compositions. *Comput. Syst. Sci. Engin. 21*, 4, 247–254.

Bagnall, A. J., Rayward-Smith, V. J., and Whittley, I . M. 2001. The next release problem. *Inf. Soft. Technol. 43*, 14, 883–890.

Baker, P., Harman, M., Steinhofel, K., and Skaliotis, A. 2006. Search based approaches to component selection and prioritization for the next release problem. In *Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM'06)*. IEEE, 176–185.

Barlas, G. and El-Fakih, K. 2008. A GA-based movie-on-demand platform using multiple distributed servers. *Multimedia Tools Appl. 40*, 3, 61–383.

Barreto, A., De O. Barros, M., and Werner, C. M. 2008. Staffing a software project: A constraint satisfaction and optimization-based approach. *Comput. Oper. Res. 35*, 10, 3073–3089.

Bate, I. and Emberson, P. 2006. Incorporating scenarios and heuristics to improve flexibility in realtime embedded systems. In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*. IEEE, 221–230.

Beyer, H. G. and Sendhoff, B. 2007. Robustness optimization—A comprehensive survey. *Comput. Meth. Appl. Mech. Engin. 196,* 3190–3218.

Bhatia, R. K., Dave, M., and Joshi, R. C. 2008. Ant colony based rule generation for reusable software component retrieval. In *Proceedings of the 1st Conference on India Software Engineering (ISEC'08)*. ACM Press, New York, 129–130.

Bodhuin, T., Canfora, G., and Troiano, L. 2007a. SORMASA: A tool for suggesting model refactoring actions by metrics-led genetic algorithm. In *Proceedings of the 1st Workshop on Refactoring Tools (WRT'07) in conjunction with ECOOP'07*. 23–24.

Bodhuin, T., Di Penta, M., and Troiano, L. 2007b. A search-based approach for dynamically repackaging downloadable applications. In *Proceedings of the Conference of the IBM Center for Advanced Studies on Collaborative Research (CASCON'07)*. ACM Press, New York, 27–41.

Bouktif, S., Antoniol, G., Merlo, E., and Neteler, M. 2006a. A novel approach to optimize clone refactoring activity. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*. ACM Press, New York, 1885–1892.

Bouktif, S., Azar, D., Precup, D., Sahraoui, H., and Kegl, B. 2004. Improving rule set based software quality prediction: A genetic algorithm-based approach. *J. Object Technol. 3,* 4, 227–241.

Bouktif, S., Kegl, B., and Sahraoui, H. 2002. Combining software quality predictive models: An evolutionary approach. In *Proceedings of the International Conference on Software Maintenance (ICSM'02).* IEEE, 385–392.

Bouktif, S., Sahraoui, H., and Antoni Ol, G. 2006b. Simulated annealing for improving software quality prediction. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06).* ACM Press, New York, 1893–1900.

Bowman, M., Briand, L. C., and Labiche, Y. 2008. Solving the class responsibility assignment problem in object-oriented analysis with multi-objective genetic algorithms. Tech. rep. SCE-07-02, Carleton University, Canada. August.

Briand, L. C., Labiche, Y., and Shousha, M. 2005. Stress testing real-time systems with genetic algorithms. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'05).* ACM Press, New York, 1021–1028.

Briand, L. C., Labiche, Y., and Shousha, M. 2006. Using genetic algorithms for early schedulability analysis and stress testing in real-time systems. *Genet. Program. Evolv. Mach. 7,* 2, 145–170.

Buhler, O. and Wegener, J. 2008. Evolutionary functional testing. *Comput. Oper. Res. 35*, 10, 3144–3160.

Burgess, C. J. and Lefley, M. 2001. Can genetic programming improve software effort estimation? A comparative evaluation. *Inf. Softw. Technol. 43,* 14, 863–873.

Burke, E. and Kendall, G. 2005. *Search Methodologies*. Introductory Tutorials in Optimization and Decision Support Techniques. Springer.

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. 2004. A lightweight approach for QoS-aware service composition. In *Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC'04).* ACM Press, New York.

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. 2005a. An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'05).* ACM Press, New York, 1069–1075.

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. 2005b. QoS-Aware replanning of composite web services. In *Proceedings of the IEEE International Conference on Web Services (ICWS'05).* IEEE, 121–129.

Cao, L, Cao, J., and Li, M. 2005a. Genetic algorithm utilized in cost-reduction driven web service selection. In *Proceedings of the International Conference on Computational Intelligence and Security (CIS'05).* Springer, 679–686.

Cao, L, Li, M., and Cao, J. 2005b. Cost-Driven web service selection using genetic algorithm. In *Proceedings of the 1st International Workshop on Internet and Network Economics (WINE'05).* Springer, 906–915.

Carlshamre, P. 2002. Release Planning in market-driven software product development: Provoking an understanding. *Requir. Engin. 7*, 3, 139–151.

Chang, C. K. 1994. Changing face of software engineering. *IEEE Softw. 11*, 1, 4–5.

Chang, C. K., Chao, C., Hsieh, S.-Y., and Alsalqan, Y. 1994. SPMNet: A formal methodology for software management. In *Proceedings of the 18th Annual International Computer Software and Applications Conference (COMPSAC'94).* IEEE, 57–57.

Chang, C. K., Chao, C., Nguyen, T. T., and Christensen, M. 1998. Software project management net: A new methodology on software management. In *Proceedings of the 22nd Annual International Computer Software and Applications Conference (COMPSAC'98)*. IEEE, 534–539.

Chang, C. K., Christensen, M. J., and Zhang, T. 2001. Genetic algorithms for project management. *Ann. Softw. Engin. 11*, 1, 107–139.

Chang, C. K., Jiang, H., Di, Y., Zhu, D., and Ge, Y. 2008. Time-Line based model for software project scheduling with genetic algorithms. *Inf. Softw. Technol. 50,* 11, 1142–1154.

Chao, C., Komada, J., Li U, Q., Muteja, M., Alsalqan, Y., and Chang, C. 1993. An application of genetic algorithms to software project management. In *Proceedings of the 9th International Advanced Science and Technology Conference*. 247–252.

Chardigny, S., Seriai, A., Oussalah, M., and Tamzalit, D. 2008a. Search-Based extraction of component-based architecture from object-oriented systems. In *Proceedings of the 2nd European Conference on Software Architecture (ECSA'08)*. Lecture Notes in Computer Science, vol. 5292. Springer, 322–325.

Chardigny, S., Seriai, A., Tamzalit, D., and Oussalah, M. 2008b. Quality-Driven extraction of a component-based architecture from an object-oriented system. In *Proceedings of the 12th European Conference on Software Maintenance and Reengineering (CSMR'08).* IEEE, 269–273.

Cheng, B. and Atlee, J. 2007. From State of the art to the future of requirements engineering. In *Future of Software Engineering*, L. Briand and A. Wolf, Eds., IEEE Computer Society.

CHICANO, F. AND ALBA, E. 2008a. Ant colony optimization with partial order reduction for discovering safety property violations in concurrent models. *Inf. Process. Lett. 106,* 6, 221–231.

CHICANO, F. AND ALBA, E. 2008b. Finding liveness errors with ACO. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI'08).* IEEE, 3002–3009.

CHICANO, F. AND ALBA, E. 2008c. Searching for liveness property violations in concurrent systems with ACO. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08)*. ACM Press, New York, 1727–1734.

CLARK, J. A. 2003. Nature-Inspired cryptography: Past, present and future. In *Proceedings of the Congress on Evolutionary Computation (CEC'03).* Vol. 3. IEEE, 1647–1654.

CLARK, J. A., DOLADO, J. J., HARMAN, M., HI ERONS, R. M., JONES, B., LUMKIN, M., MITCHELL, B., MANCORIDIS, S., REES, K., ROPER, M., AND SHEPPERD, M. J. 2003. Reformulating software engineering as a search problem. *IEE Proc. Softw. 150*, 3, 161–175.

CLARK, J. A., HARMAN, M., HIERONS, R. M., JONES, B., LUMKIN, M., REES, K., ROPER, M., AND SHEPPERD, M. J. 2000. The application of metaheuristic search techniques to problems in software engineering. Tech. rep. TR-01-2000, University of York, Brunel University, University of Glamorgan, British Telecom, Strathclyde University, Bournemouth University. August.

CLARK, J. A. AND JACOB, J. L. 2000. Searching for a solution: Engineering tradeoffs and the evolution of provably secure protocols. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'00).* IEEE, 82–95.

CLARK, J. A. AND JACOB, J. L. 2001. Protocols are programs too: The meta-heuristic search for security protocols. *Inf. Softw. Technol. 43*, 14, 891–904.

COFFMAN, E. J., GAREY, M., AND JOHNSON, D. 1984. Approximation algorithms for bin-packing. In *Algorithm Design for Computer System Design*. Springer.

COHEN, M. B., KOOI, S. B., AND SRISAAN, W. 2006. Clustering the heap in multi-threaded applications for improved garbage collection. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06).* ACM Press, New York, 1901–1908.

COLLETTE, Y. AND SIARRY, P. 2004. *Multiobjective Optimisation: Principles and Case Studies*. Springer.

COOPER, K. D., SCHIELKE, P. J., AND SUBRAMANIAN, D. 1999. Optimizing for reduced code space using genetic algorithms. In *Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers and Tools for Embedded Systems (LCTES'99).* ACM Press, New York, 1–9.

CORTELLESSA, V., CRNKOVIC, I., MARINELLI, F., AND POTENA, P. 2008a. Experimenting the automated selection of COTS components based on cost and system requirements. *J. Univers. Comput. Sci. 14*, 8, 1228–1255.

CORTELLESSA, V., MARINELLI, F., AND POTENA, P. 2006. Automated selection of software components based on cost/reliability tradeoff. In *Proceedings of the 3rd European Workshop on Software Architecture (EWSA'06).* Lecture Notes in Computer Science, vol. 4344. Springer, 66–81.

CORTELLESSA, V., MARINELLI, F., AND POTENA, P. 2008b. An optimization framework for "build-orbuy" decisions in software architecture. *Comput. Oper. Res. 35*, 10, 3090–3106.

COWAN, G. S., REYNOLDS, R. G., AND COWAN, G. J. 2004. *Acquisition of Software Engineering Knowledge SWEEP: An Automatic Programming System based on Genetic Programming and Cultural Algorithms.* Software Engineering and Knowledge Engineering Series, vol. 14. World Scientific.

DEL GROSSO, C., ANTONIOL, G., DI PENTA, M., GALINIER, P., AND MERLO, E. 2005. Improving network applications security: A new heuristic to generate stress testing data. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'05).* ACM Press, New York, 1037–1043.

DEL GROSSO, C., ANTONIOL, G., MERLO, E., AND GALINIER, P. 2008. Detecting buffer overflow via automatic test input data generation. *Comput. Oper. Res. 35*, 10, 3125–3143.

DEL ROSSO, C. 2006. Reducing Internal Fragmentation in segregated free lists using genetic algorithms. In *Proceedings of the International Workshop on Interdisciplinary Software Engineering Research (WISER'06).* ACM Press, New York, 57–60.

DESNOS, N., HUCHARD, M., TREMBLAY, G., URTADO, C., AND VAUTTIER, S. 2008. Search-Based many-to-one component substitution. *J. Softw. Maint. Evol. Res. Pract. 20*, 5, 321–344.

DI PENTA, M. 2005. Evolution Doctor: A framework to control software system evolution. In *Proceedings of the 9th European Conference on Software Maintenance and Reengineering (CSMR'05)*. IEEE, 280–283.

DI PENTA, M., LOMBARDI, P., TANEJA, K., AND TROIANO, L. 2008. Search-Based inference of dialect grammars. *Soft Comput. Fus. Found. Meth. Appl. 12,* 1, 51–66.

DI PENTA, M., NETELER, M., ANTONIOL, G., AND MERLO, E. 2005. A language-independent software renovation framework. *J. Syst. Softw. 77*, 3, 225–240.

DI PENTA, M. AND TANEJA, K. 2005. Towards the automatic evolution of reengineering tools. In *Proceedings of the 9th European Conference on Software Maintenance and Reengineering (CSMR'05)*. IEEE, 241–244.

DILLON, E. 2005. Hybrid approach for the automatic determination of worst case execution time for embedded systems written in C. M.S. thesis, Institute of Technology, Carlow, Ireland.

DOLADO, J. J. 2000. A validation of the component-based method for software size estimation. *IEEE Trans. Softw. Engin. 26*, 10, 1006–1021.

DOLADO, J. J. 2001. On the problem of the software cost function. *Inf. Softw. Technol. 43*, 1, 61–72.

DOLADO, J. J. AND FERNANDEZ, L. 1998. Genetic programming, neural networks and linear regression in software project estimation. In *Proceedings of International Conference on Software Process Improvement, Research, Education and Training (INSPIREIII)*. British Computer Society, 157–171.

DORIGO, M. AND BLUM, C. 2005. Ant colony optimization theory: A survey. *Theor. Comput. Sci. 344*, 2-3, 243–278.

DOVAL, D., MANCORIDIS, S., AND MITCHELL, B. S. 1999. Automatic clustering of software systems using a genetic algorithm. In *Proceedings of the International Conference on Software Tools and Engineering Practice (STEP'99)*. IEEE, 73–81.

DOZIER, G., BROWN, D., HOU, H., AND HURLEY, J. 2007. Vulnerability analysis of immunity-based intrusion detection systems using evolutionary hackers. *Appl. Soft Comput. 7*, 2, 547–553.

DOZIER, G., BROWN, D., HURLEY, J., AND CAIN, K. 2004. Vulnerability analysis of immunity-based intrusion detection systems using evolutionary hackers. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'04)*. Lecture Notes in Computer Science, vol. 3102. Springer, 263–274.

DUBACH, C., CAVAZ OS, J., FRANKE, B., O'BOYL, E., M., FURSIN, G., AND TEMAM, O. 2007. Fast compiler optimisation evaluation using code-feature based performance prediction. In *Proceedings of the 4th International Conference on Computing Frontiers.*ACM Press, New York, 131–142.

EL-FAKI H, K., YAMAGUCHI, H., AND V. BOCHMANN, G. 1999. A method and a genetic algorithm for deriving protocols for distributed applications with minimum communication cost. In *Proceedings of the 11th International Conference on Parallel and Distributed Computing and Systems (PDCS'99)*.

EMBERSON, P. AND BATE, I. 2007. Minimising task migration and priority changes in mode transitions. In *Proceedings of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'07)*. IEEE, 158–167.

ERNST, M. D. 2000. Dynamically discovering likely program invariants. Ph.D. thesis, University of Washington.

ERNST, M. D., COCKRELL, J., GRISWOLD, W. G., AND NOTKIN, D. 2001. Dynamically discovering likely program invariants to support program evolution. *IEEE Trans. Softw. Engin. 27*, 2, 1–25.

EVERSON, R. M. AND FIELDSEND, J. E. 2006. Multiobjective optimization of safety related systems: An application to short-term conflict alert. *IEEE Trans. Evol. Comput. 10*, 2, 187–198.

EVETT, M. P., KHOSHGOFTAAR, T. M. , DER CHIEN, P., AND ALLEN, E. B. 1999. Using genetic programming to determine software quality. In *Proceedings of the 12th International Florida Artificial Intelligence Research Society Conference (FLAIRS'99)*. Florida Research Society, 113–117.

FATIREGUN, D., HARMAN, M., AND HIERONS, R. 2003. Search based transformations. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'03)*. Lecture Notes in Computer Science, vol. 2724. Springer, 2511–2512.

FATIREGUN, D., HARMAN, M., AND HIERONS, R. 2005. Search-Based amorphous slicing. In *Proceedings of the 12th International Working Conference on Reverse Engineering (WCRE'05)*. IEEE, 3–12.

FATIREGUN, D., HARMAN, M., AND HIERONS, R. M. 2004. Evolving transformation sequences using genetic algorithms. In *Proceedings of the 4th IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'04)*. IEEE, 65–74.

FEATHER, M. S., CORNFORD, S. L., KIPER, J. D., AND MENZIES, T. 2006. Experiences using visualization techniques to present requirements, risks to them, and options for risk mitigation. In *Proceedings of the International Workshop on Requirements Engineering Visualization (REV'06)*. IEEE, 10.

FEATHER, M. S., KIPER, J. D., AND KALAFAT, S. 2004. Combining heuristic search, visualization and data mining for exploration of system design space. In *Proceedings of the 14th Annual International Symposium of the Council on Systems Engineering (INCOSE'04)*.

FEATHER, M. S. AND MENZIES, T. 2002. Converging on the optimal attainment of requirements. In *Proceedings of the 10th IEEE International Conference on Requirements Engineering (RE'02)*. IEEE, 263–270.

FELDT, R. 1998a. An experiment on using genetic programming to develop multiple diverse software variants. Tech. rep. 98-13, Chalmers University of Technology, Gothenburg, Sweden. September.

FELDT, R. 1998b. Generating Multiple diverse software versions with genetic programming. In *Proceedings of the 24th EUROMICRO Conference (EUROMICRO'98)*. Vol. 1. IEEE, 387–394.

FELDT, R. 1998c. Generating multiple diverse software versions with genetic programming - An experimental study. *IEE Proc. Softw. 145*, 6, 228–236.

FELDT, R. 1999. Genetic programming as an explorative tool in early software development phases. In *Proceedings of the 1st International Workshop on Soft Computing Applied to Software Engineering (SCASE'99),* C. Ryan and J. Buckley, Eds. 11–20.

FELDT, R. 2002. An interactive software development workbench based on biomimetic algorithms. Tech. rep. 02-16, Chalmers University of Technology, Gothenburg, Sweden. November.

FELDT, R., TORKAR, R., GORSCHEK, T., AND AFZAL, W. 2008. Searching for cognitively diverse tests: Test variability and test diversity metrics. In *Proceedings of 1st International Workshop on Search-Based Software Testing (SBST) in conjunction with ICST'08*. IEEE, 178–186.

FERREIRA, M., CHI CANO, F., ALBA, E., AND GOMEZ-PULIDO, J. A. 2008. Detecting protocol errors using particle swarm optimization with Java pathfinder. In *Proceedings of the High Performance Computing and Simulation Conference (HPCS'08),* W. W. Smari, Ed. 319–325.

FINKELSTEIN, A., HARMAN, M., MANS OURI, S. A., REN, J., AND ZHANG, Y. 2008. "Fairness analysis" in requirements assignments. In *Proceedings of the 16th IEEE International Requirements Engineering Conference (RE'08)*. IEEE, 115–124.

FUNES, P., BONABEAU, E., HERVE, J., AND MORI EUX, Y. 2004. Interactive multi-participant task allocation. In *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 1699–1705.

GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. 1995. *Design Patterns*. Addison-Wesley.

GAROUSI, V. 2006. Traffic-Aware stress testing of distributed real-time systems based on UML models using genetic algorithms. Ph.D. thesis, Carleton University, Canada.

GAROUSI, V. 2008. Empirical analysis of a genetic algorithm-based stress test technique. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08).* ACM Press, New York, 1743–1750.

GAROUSI, V., BRIAND, L. C., AND LABICHE, Y. 2008. Traffic-Aware stress testing of distributed realtime systems based on UML models using genetic algorithms. *J. Syst. Softw. 81*, 2, 161–185.

GODEFROID, P. 1997. Model checking for programming languages using verisoft. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'97).* ACM Press, New York, 174–186.

GOLD, N., HARMAN, M., LI, Z., AND MAHDAVI, K. 2006. Allowing overlapping boundaries in source code using a search based approach to concept binding. In *Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM'06).* IEEE, 310–319.

GOLDSBY, H. J. AND CHENG, B. H. 2008a. Automatically generating behavioral models of adaptive systems to address uncertainty. In *Proceedings of the 11th International Conference on Model Driven Engineering Languages and Systems (MoDELS'08).* Springer, 568–583.

GOLDSBY, H. J. AND CHENG, B. H. 2008b. Avida-MDE: A digital evolution approach to generating models of adaptive software behavior. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08).* ACM Press, New York, 1751–1758.

GOLDSBY, H. J., CHENG, B. H., MCKINLEY, P. K., KNOESTER, D. B., AND OFRIA, C. A. 2008. Digital evolution of behavioral models for autonomic systems. In *Proceedings of the International Conference on Autonomic Computing (ICAC'08)*. IEEE, 87–96.

GREER, D. AND RUHE, G. 2004. Software release planning: An evolutionary and iterative approach. *Inf. Softw. Technol. 46*, 4, 243–253.

GROSS, H.-G. 2000. Measuring evolutionary testability of real-time software. Ph.D. thesis, University of Glamorgan, UK.

GROSS, H.-G. 2001. A prediction system for evolutionary testability applied to dynamic execution time analysis. *Inf. Softw. Technol. 43*, 14, 855–862.

GROSS, H.-G., JONES, B. F., AND EYRES, D. E. 2000. Structural performance measure of evolutionary testing applied to worst-case timing of real-time systems. *IEE Proc. Softw. 147*, 2, 25–30.

GROSS, H.-G. AND MAYER, N. 2002. Evolutionary testing in component-based real-time system con-struction. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'02).* Morgan Kaufmann, 207–214.

GROSS, H.-G. AND MAYER, N. 2003. Search-Based execution-time verification in object-oriented and component-based real-time system development. In *Proceedings of the 8th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'03).* IEEE, 113–120.

HAAS, J., PEYSAKHOV, M., AND MANCORIDIS, S. 2005. GA-Based parameter tuning for multi-agent systems. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'05).* ACM Press, New York, 1085–1086.

HARMAN, M. 2006. Search-Based software engineering for maintenance and reengineering. In *Proceedings of the 10th European Conference on Software Maintenance and Reengineering (CSMR'06)*. IEEE, 311.

HARMAN, M. 2007a. Search based software engineering for program comprehension. In *Proceedings of the 15th IEEE International Conference on Program Comprehension (ICPC'07)*. IEEE, 3–13.

HARMAN, M. 2007b. The current state and future of search based software engineering. In *Proceedings of the International Conference on Software Engineering/Future of Software Engineering (ICSE/FOSE'07)*. IEEE, 342–357.

HARMAN, M. 2010. Why the virtual nature of software makes it ideal for search based optimization. In *Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering (FASE'10)*. Springer, 1–12.

HARMAN, M. AND CLARK, J. A. 2004. Metrics are fitness functions too. In *Proceedings of the 10th IEEE International Symposium on Software Metrics (METRICS'04)*. IEEE, 58–69.

HARMAN, M., HASSOUN, Y., LAKHOTIA, K., MCMINN, P., AND WEGENER, J. 2007a. The impact of input domain reduction on search-based test data generation. In *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*. ACM Press, New York, 155–164.

HARMAN, M., HIERONS, R., AND PROCTOR, M. 2002. A new representation and crossover operator for search-based optimization of software modularization. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'02)*. Morgan Kaufmann, 1351–1358.

HARMAN, M. AND JONES, B. F. 2001a. Search-Based software engineering. *Inf. Softw. Technol. 43*, 14, 833–839.

HARMAN, M. AND JONES, B. F. 2001b. Software engineering using metaheuristic innovative algorithms: Workshop report. *Inf. Softw. Technol. 43*, 14, 762–763.

HARMAN, M. AND JONES, B. F. 2001c. The SEMINAL workshop: Reformulating software engineering as a metaheuristic search problem. *ACM SIGSOFT Softw. Engin. Not. 26*, 6, 62–66.

HARMAN, M., LAKHOTIA, K., AND MCMINN, P. 2007b. A multi-objective approach to search-based test data generation. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*. ACM Press, New York, 1098–1105.

HARMAN, M., MANSOURI, A., AND ZHANG, Y. 2009. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Tech. rep. TR-09-03, Department of Computer Science, King's College London. April.

HARMAN, M., SKALIOTIS, A., AND STEINHOFEL, K. 2006. Search-Based approaches to the component selection and prioritization problem. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*. ACM Press, New York, 1951–1952.

HARMAN, M., SWIFT, S., AND MAHDAVI, K. 2005. An empirical study of the robustness of two module clustering fitness functions. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'05)*. Vol. 1. ACM Press, New York, 1029–1036.

HARMAN, M. AND TRATT, L. 2007. Pareto optimal search based refactoring at the design level. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*. ACM Press, New York, 1106–1113.

HARMAN, M. AND WEGENER, J. 2004. Getting results from search-based approaches to software engi-neering. In *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*. IEEE, 728–729.

HART, J. AND SHEPPERD, M. J. 2002. Evolving software with multiple outputs and multiple populations. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'02)*. Morgan Kaufmann, 223–227.

HE, P., KANG, L, AND FU, M. 2008. Formality based genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'08) IEEE World Congress on Computational Intelligence)*. IEEE, 4080–4087.

HERICKO, M., ZIVKOVI C, A., AND ROZMAN, I. 2008. An approach to optimizing software development team size. *Inf. Process. Lett. 108*, 3, 101–106.

HODJAT, B., ITO, J., AND AMAMIYA, M. 2004. A genetic algorithm to improve agent-oriented natural language interpreters. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'04)*. Lecture Notes in Computer Science, vol. 3103. Springer, 1307–1309.

HOSTE, K. AND EECKHOUT, L. 2008. Cole: Compiler optimization level exploration. In *Proceedings of the 6th Annual IEEE/ACM International Symposium on Code Generation and Optimization (CGO'08)*. ACM Press, New York, 165–174.

HSU, C.-J., HUANG, C.-Y., AND CHEN, T.-Y. 2008. A modified genetic algorithm for parameter estimation of software reliability growth models. In *Proceedings of the 19th International Symposium on Software Reliability Engineering (ISSRE'08)*. IEEE, 281–282.

HUANG, S.-J., CHIU, N.-H., AND CHEN, L.-W. 2008. Integration of the grey relational analysis with genetic algorithm for software effort estimation. Euro. *J. Oper. Res. 188*, 3, 898–909.

HUYNH, S. AND CAI, Y. 2007. An evolutionary approach to software modularity analysis. In *Proceedings of the 1st International Workshop on Assessment of Contemporary Modularization Techniques (ACoM'07)*. ACM Press, New York, 1–6.

JAEGER, M. C. AND MUHL, G. 2007. QoS-Based selection of services: The implementation of a genetic algorithm. In *Proceedings of Kommunikation in Verteilten Systemen (KiVS'07) Workshop: ServiceOriented Architectures and Service-Oriented Computing.*

JALALI, O., MENZIES, T., AND FEATHER, M. 2008. Optimizing requirements decisions with KEYS. In *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering (PROMISE'08).* ACM Press, New York, 79–86.

JARILLO, G., SUCCI, G., PEDRYCZ, W., AND REFORMAT, M. 2001. Analysis of software engineering data using computational intelligence techniques. In *Proceedings of the 7th International Conference on Object Oriented Information Systems (OOIS'01).* Springer, 133–142.

JIANG, H. 2006. Can the genetic algorithm be a good tool for software engineering searching problems? In *Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)*. IEEE, 362–366.

JIANG, H., CHANG, C. K., ZHU, D., AND CHENG, S. 2007a. A foundational study on the applicability of genetic algorithm to software engineering problems. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC'07)*. IEEE, 2210–2219.

JIANG, T., GOLD, N., HARMAN, M., AND LI, Z. 2007b. Locating dependence structures using search-based slicing. *Inf. Softw. Technol. 50*, 12, 1189–1209.

JIANG, T., HARMAN, M., AND HASSOUN, Y. 2008. Analysis of procedure splitability. In *Proceedings of the 15th Working Conference on Reverse Engineering (WCRE'08).* IEEE, 247–256.

JOHNSON, C. 2007. Genetic programming with fitness based on model checking. In *Proceedings of the 10th European Conference on Genetic Programming (EuroGP'07)*. Lecture Notes in Computer Science, vol. 4445. Springer, 114–124.

JOSHI, A. M., EECKHOUT, L., JOHN, L. K., AND ISEN, C. 2008. Automated microprocessor stressmark generation. In *Proceedings of the 14th IEEE International Symposium on High Performance Computer Architecture (HPCA'08).* IEEE, 229–239.

KAPUR, P., NGO-THE, A., RUHE, G., AND SMITH, A. 2008. Optimized staffing for product releases and its application at Chartwell Technology. *J. Softw. Maint. Evol. Res. Pract. 20,* 5, 365–386.

KARLSSON, J., WOHLIN, C., AND REGNELL, B. 1998. An evaluation of methods for priorizing software requirements. *Inf. Softw. Technol. 39*, 939–947.

KATZ, G. AND PELED, D. 2008a. Genetic programming and model checking: Synthesizing new mutual exclusion algorithms. In *Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis (ATVA'08).* Lecture Notes in Computer Science, vol. 5311. Springer, 33–47.

KATZ, G. AND PELED, D. 2008b. Model checking-based genetic programming with an application to mutual exclusion. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08).* Springer, 141–156.

KESSENTINI, M., SAHRAOUI, H., AND BOUKADOUM, M. 2008. Model transformation as an optimization problem. In *Proceedings of the ACM/IEEE 11th International Conference on Model Driven Engineering Languages and Systems (MODELS'08).* Lecture Notes in Computer Science, vol. 5301. Springer, 159-173.

KHOSHGOFTAAR, T. M. AND LIU, Y. 2007. A multi-objective software quality classification model using genetic programming. *IEEE Trans. Reliab. 56*, 2, 237–245.

KHOSHGOFTAAR, T. M., LIU, Y., AND SELIYA, N. 2003. Genetic programming-based decision trees for software quality classification. In *Proceedings of the 15th International Conference on Tools with Articial Intelligence (ICTAI'03).* IEEE, 374–383.

KHOSHGOFTAAR, T. M., LIU, Y., AND SELIYA, N. 2004a. A multiobjective module-order model for software quality enhancement. *IEEE Trans. Evol. Comput.* 8, 6, 593–608.

KHOSHGOFTAAR, T. M., LIU, Y., AND SELIYA, N. 2004b. Module-Order modeling using an evolutionary multiobjective optimization approach. In *Proceedings of the 10th IEEE International Symposium on Software Metrics (METRICS'04).* IEEE, 159–169.

KHOSHGOFTAAR, T. M., SELIYA, N., AND DROWN, D. J. 2008. On the rarity of fault-prone modules in knowledge-based software quality modeling. In *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE'08).* 279–284.

KIPER, J. D., FEATHER, M. S., AND RICHARDSON, J. 2007. Optimizing the v&v process for critical systems. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07).* ACM Press, New York, 1139–1139.

KIRSOPP, C., SHEPPERD, M. J., AND HART, J. 2002. Search heuristics, case-based reasoning and software project effort prediction. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'02)*. Morgan Kaufmann, 1367–1374.

KUPERBERG, M., KROGMANN, K., AND REUSSNER, R. 2008. Performance prediction for black-box com- ponents using reengineered parametric behaviour models. In *Proceedings of the 11th International Symposium on Component-Based Software Engineering (CBSE'08)*. Lecture Notes in Computer Science, vol. 5282. Springer, 48–63.

LAKHOTIA, K., TILLMANN, N., HARMAN, M. , AND DE HALLEUX, J. 2010. FloPSy - Search-Based floating point constraint solving for symbolic execution. In *Proceedings of the 22nd IFIP International Conference on Testing Software and Systems (ICTSS'10)*. Lecture Notes in Computer Science, vol. 6435. Springer, 142–157.

LANGE, R. AND MANCORIDIS, S. 2007. Using code metric histograms and genetic algorithms to perform author identification for software forensics. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*. ACM Press, New York, 2082–2089.

LEFLEY, M. AND SHEPPERD, M. J. 2003. Using genetic programming to improve software effort estimation based on general data sets. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'03)*. Lecture Notes in Computer Science, vol. 2724. Springer, 2477–2487.

LI, C., VAN DEN AKKER, M., BRINKKEMP ER, S., AND DIEPEN, G. 2007. Integrated requirement selection and scheduling for the release planning of a software product. In *Proceedings of the 13th International Working Conference on Requirements Engineering: Foundation for Software Quality (RefsQ'07)*. Lecture Notes in Computer Science, vol. 4542. Springer, 93–108.

LINDEN, D. S. 2002. Innovative antenna design using genetic algorithms. In *Creative Evolutionary Sys-tems*, D. W. Corne and P. J. Bentley, Eds. Elsevier, Chapter 20.

LIU, Y. AND KHOSHGOFTAAR, T. 2004. Reducing overfitting in genetic programming models for software quality classification. In *Proceedings of the 8th IEEE International Symposium on High Assurance Systems Engineering (HASE'04)*. IEEE, 56–65.

LIU, Y. AND KHOSHGOFTAAR, T. M. 2001. Genetic programming model for software quality classification. In *Proceedings of the 6th IEEE International Symposium on High-Assurance Systems Engineering: Special Topic: Impact of Networking (HASE'01)*. IEEE, 127–136.

LIU, Y. AND KHOSHGOFTAAR, T. M. 2003. Building decision tree software quality classification models using genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03)*. Lecture Notes in Computer Science, vol. 2724. Springer, 1808–1809.

LOKAN, C. 2005. What should you optimize when building an estimation model? In *Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS'05)*. IEEE, 34–44.

LUCAS, S. M. AND REYNOLDS, T. J. 2005. Learning deterministic finite automata with a smart state labeling evolutionary algorithm. *IEEE Trans. Pattern Anal. Mach. Intell. 27*, 7, 1063–1074.

LUTZ, R. 2001. Evolving good hierarchical decompositions of complex systems. *J. Syst. Archit. 47*, 7, 613–634.

MA, Y. AND ZHANG, C. 2008. Quick convergence of genetic algorithm for QoS-driven web service selection. *Comput. Netw. 52*, 5, 1093–1104.

MAHANTI, P. K. AND BANERJEE, S. 2006. Automated testing in software engineering: Using ant colony and self-regulated swarms. In *Proceedings of the 17th IASTED International Conference on Modelling and Simulation (MS'06)*. 443–448.

MAHDAVI , K. 2005. A Clustering genetic algorithm for software modularisation with multiple hill climbing approach. Ph.D. thesis, Brunel University West London, UK.

MAHDAVI, K., HARMAN, M., AND HIERONS, R. 2003a. Finding building blocks for software clustering. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'03)*. Lecture Notes in Computer Science, vol. 2724. Springer, 2513–2514.

MAHDAVI, K., HARMAN, M., AND HIERONS, R. M. 2003b. A multiple hill climbing approach to software module clustering. In *Proceedings of the International Conference on Software Maintenance (ICSM'03)*. IEEE, 315–324.

MANCORIDIS, S., MITCHELL, B. S., CHEN, Y., AND GANSNER, E. R. 1999. Bunch: A clustering tool for the recovery and maintenance of software system structures. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM'99)*. IEEE, 50–59.

MANCORIDIS, S., MITCHELL, B. S., RORRES, C., CHEN, Y., AND GANSNER, E. R. 1998. Using automatic clustering to produce high-level system organizations of source code. In *Proceedings of the 6th Inter-national Workshop on Program Comprehension (IWPC'98)*. IEEE, 45–52.

MANTERE, T. 2003. Automatic software testing by genetic algorithms. Ph.D. thesis, University of Vaasa, Finland.

MANTERE, T. AND ALANDER, J. T. 2005. Evolutionary software engineering, a review. *Appl. Soft. Comput. 5*, 3, 315–331.

MCMINN, P. 2004. Search-Based software test data generation: A survey. *Softw. Test. Verif. Reliab. 14,* 2, 105–156.

MILLER, W. AND SPOONER, D. L. 1976. Automatic generation of floating-point test data. *IEEE Trans. Softw. Engin. 2*, 3, 223–226.

MINOHARA, T. AND TOHMA, Y. 1995. Parameter estimation of hyper-geometric distribution software reliability growth model by genetic algorithms. In *Proceedings of the 6th International Symposium on Software Reliability Engineering (ISSRE'95).* IEEE, 324–329.

MITCHELL, B. S. 2002. A heuristic search approach to solving the software clustering problem. Ph.D. thesis, Drexel University.

MITCHELL, B. S. AND MANCORIDIS, S. 2002. Using heuristic search techniques to extract design ab-stractions from source code. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'02)*. Morgan Kaufmann, 1375–1382.

MITCHELL, B. S. AND MANCORIDIS, S. 2003. Modeling the search landscape of metaheuristic software clustering algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03)* Springer, 2499–2510.

MITCHELL, B. S. AND MANCORIDIS, S. 2006. On the automatic modularization of software systems using the bunch tool. *IEEE Trans. Softw. Engin. 32*, 3, 193–208.

MITCHELL, B. S. AND MANCORIDIS, S. 2008. On the evaluation of the bunch search-based software modularization algorithm. *Soft Comput. Fus. Found. Meth. Appl. 12*, 1, 77–93.

MITCHELL, B. S., MANCORIDIS, S., AND TRAVERSO, M. 2002. Search based reverse engineering. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02).* ACM Press, New York, 431–438.

MITCHELL, B. S., MANCORIDIS, S., AND TRAVERSO, M. 2004. Using interconnection style rules to infersoftware architecture relations. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'04)*. Lecture Notes in Computer Science, vol. 3103. Springer, 1375–1387.

MITCHELL, B. S., TRAVERSO, M. AND MANCORIDIS, S. 2001. An architecture for distributing the computation of software clustering algorithms. In *Proceedings of the IEEE/IFIP Working Conference on Software Architecture (WICSA'01)*. IEEE Computer Society, 181–190.

MONNIER, Y., BEAUVAIS, J.-P., AND DEPLANCHE, A.-M. 1998. A genetic algorithm for scheduling tasks in a real-time distributed system. In *Proceedings of the 24th EUROMICRO Conference (EUROMICRO'98).* Vol. 2. IEEE, 20708–20714.

NGUYEN, C., MILES, S., PERINI, A., TONELLA, P., HARMAN, M., AND LUCK, M. 2009. Evolutionary testing of autonomous software agents. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09).* 521–528.

NISBET, A. 1998. Gaps: A compiler framework for genetic algorithm (GA) optimised parallelisation. In *Proceedings of the International Conference and Exhibition on High-Performance Computing and Networking (HPCN'98).* Lecture Notes in Computer Science, vol. 1401. Springer, 987–989.

O'KEEFFE, M. AND O CINNEIDE, M. 2003. A stochastic approach to automated design improvement. In *Proceedings of the 2nd International Conference on Principles and Practice of Programming in Java (PP-PJ'03).* Computer Science Press, 59–62.

O'KEEFFE, M. AND O CINNEIDE, M. 2004. Towards automated design improvement through combinatorial optimisation. In *Proceedings of the 26th International Conference on Software Engineering and Workshop on Directions in Software Engineering Environments (WoDiSEE'04).* ACM Press, New York, 75–82.

O'KEEFFE, M. AND O CINNEIDE, M. 2006. Search-Based software maintenance. In *Proceedings of the Conference on Software Maintenance and Reengineering (CSMR'06).* IEEE, 249–260.

O'KEEFFE, M. AND O CINNEIDE, M. 2007. Getting the most from search-based refactoring. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07).* ACM Press, New York, 1114–1120.

O'KEEFFE, M. AND O CINNEIDE, M. 2008a. Search-Based refactoring: An empirical study. *J. Softw. Maint. Evol. Res. Pract. 20*, 5, 345–364.

O'KEEFFE, M. AND O CINNEIDE, M. 2008b. Search-Based refactoring for software maintenance. *J. Syst. Softw. 81*, 4, 502–516.

PEDRYCZ, W. 2002. Computational intelligence as an emerging paradigm of software engineering. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE'02).* ACM Press, New York, 7–14.

POHLHEIM, H. AND WEGENER, J. 1999. Testing the temporal behavior of real-time software modules using extended evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)*. Vol. 2. Morgan Kaufmann, 1795.

POULDING, S., EMBERSON, P., BATE, I., AND CLARK, J. A. 2007. An efficient experimental methodology for configuring search-based design algorithms. In *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE'07)*. IEEE, 53–62.

PRADITWONG, K., HARMAN, M., AND YAO, X. 2010. Software module clustering as a multi-objective search problem. *IEEE Trans. Softw. Engin. 37*, 2, 264–282.

RAIHA, O. 2008a. Applying genetic algorithms in software architecture design. M.S. thesis, University of Tampere, Finland.

RAIHA, O. 2008b. Phil. lic. thesis. Ph.D. thesis, University of Tampere, Finland.

RAIHA, O. 2010. A survey on search-based software designs. *Comput. Sci. Rev. 4*, 4, 203–249.

RAIHA, O., KOSKIMIES, K., AND MAKINEN, E. 2008a. Genetic synthesis of software architecture. In *Proceedings of the 7th International Conference on Simulated Evolution and Learning (SEAL'08)*. Lecture Notes in Computer Science, vol. 5361. Springer, 565–574.

RAIHA, O., KOSKIMIES, K., MAKINEN, E., AND SYSTA, T. 2008b. Pattern-Based genetic model refinements in MDA. *Nordic J. Comput. 14*, 4, 338–355.

ERFORMAT, M., CHAI, X., AND MILLER, J. 2003. Experiments in automatic programming for general pur-poses. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03)*. IEEE, 366–373.

REFORMAT, M., CHAI, X., AND MILLER, J. 2007. On the possibilities of (pseudo-) software cloning from external interactions. *Soft Comput. Fus. Found. Meth. Appl. 12*, 1, 29–49.

RELA, L. 2004. Evolutionary Computing In search-based software engineering. M.S. thesis, Lappeenranta University of Technology, Finland.

REN, J., HARMAN, M., AND DI PENTA, M. 2011. Cooperative co-evolutionary optimization of software project staff assignments and job scheduling. In *Proceedings of the 3rd International Symposium on Search Based Software Engineering (SSBSE'11)*. Springer.

RUHE, G. AND GREER, D. 2003. Quantitative studies in software release planning under risk and resource constraints. In *Proceedings of the International Symposium on Empirical Software Engineering (ISESE'03)*. IEEE, 262–270.

RUHE, G. AND NGO-THE, A. 2004. Hybrid intelligence in software release planning. *Int. J. Hybrid Intell. Syst. 1*, 1-2, 99–110.

RUHE, G. AND SALIU, M. O. 2005. The art and science of software release planning. *IEEE Softw. 22*, 6, 47–53.

RYAN, C. 2000. *Automatic Re-Engineering of Software Using Genetic Programming*. Vol. 2. Kluwer Academic Publishers.

SAHRAOUI, H. A., VALTCHEV, P., KONKOBO, I., AND SHEN, S. 2002. Object identification in legacy code as a grouping problem. In *Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment (COMPSAC'02)*. IEEE, 689–696.

SALIU, M. O. AND RUHE, G. 2007. Bi-Objective release planning for evolving software systems. In *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*. ACM Press, New York, 105–114.

SCHNIER, T., YAO, X., AND LIU, P. 2004. Digital filter design using multiple pareto fronts. *Soft Comput. 8*, 5, 332–343.

SENG, O., BAUER, M., BIEHL, M., AND PACHE, G. 2005. Search-Based improvement of subsystem decompositions. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'05)*. ACM Press, New York, 1045–1051.

SENG, O., STAMMEL, J., AND BURKHART, D. 2006. Search-Based determination of refactorings for improving the class structure of object-oriented systems. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*. ACM Press, New York, 1909–1916.

SHAN, Y., MCKAY, R. I., LOKAN, C. J., AND ESSAM, D. L. 2002. Software project effort estimation using genetic programming. In *Proceedings of the IEEE International Conference on Communications, Circuits and Systems and West Sino Expositions (ICCCSWSE'02)*. Vol. 2. IEEE, 1108–1112.

SHANNON, C. E. 1948. A mathematical theory of communication. *Bell Syst. Tech. J. 27*, 379–423 and 623–656.

SHARMA, V. S. AND JALOTE, P. 2008. Deploying software components for performance. In *Proceedings of the 11th International Symposium on Component-Based Software Engineering (CBSE'08)*. Springer, 32–47.

SHEPPERD, M. 2007. Software economics. In *Future of Software Engineering*, L. Briand and A. Wolf, Eds., IEEE Computer Society.

SHEPPERD, M. J. 1995. *Foundations of Software Measurement*. Prentice Hall PTR.

SHETA, A. F. 2006. Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects. *J. Comput. Sci. 2*, 2, 118–123.

SHEU, S.-T. AND CHUANG, Y.-R. 2006. A pipeline-based genetic algorithm accelerator for time-critical processes in real-time systems. *IEEE Trans. Comput. 55*, 11, 1435–1448.

SHUKLA, K. K. 2000. Neuro-Genetic prediction of software development effort. *Inf. Softw. Technol. 42*, 10, 701–713.

SHYANG, W., LAKOS, C., MI CHALEWICZ, Z., AND SCHELLENBERG, S. 2008. Experiments in applying evolutionary algorithms to software verification. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC'08)*. IEEE, 3531–3536.

SIMONS, C. L. AND PARMEE, I. C. 2006. Single and multi-objective genetic operators in object-oriented conceptual software design. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*. ACM Press, New York, 1957–1958.

SIMONS, C. L. AND PARMEE, I. C. 2007. A cross-disciplinary technology transfer for search-based evolutionary computing: From engineering design to software engineering design. *Engin. Optim. 39*, 5, 631–648.

SIMONS, C. L. AND PARMEE, I. C. 2008a. Agent-Based support for interactive search in conceptual software engineering design. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08)*. ACM Press, New York, 1785–1786.

SIMONS, C. L. AND PARMEE, I. C. 2008b. User-Centered, evolutionary search in conceptual software design. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'08) World Congress on Computational Intelligence*. IEEE, 869–876.

SINCLAIR, M. C. AND SHAMI, S . H. 1997. Evolving simple software agents: Comparing genetic algorithm and genetic programming performance. In *Proceedings of the 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97)*. IEE, 421–426.

SNELTING, G. 1998. Concept analysis—A new framework for program understanding. *ACM SIGPLAN Not. 33*, 1–10.

STEPHENSON, M., AMARASINGHE, S., MARTIN, M., AND O'REILLY, U.-M. 2003a. Meta optimization: Improving compiler heuristics with machine learning. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'03)*. ACM Press, New York, 77–90.

STEPHENSON, M., O'REILLY, U.-M., MARTIN, M. C., AND AMARASINGHE, S. 2003b. Genetic programming applied to compiler heuristic optimization. In *Proceedings of the 6th European Conference on Genetic Programming (EuroGP'03)*. Lecture Notes in Computer Science, vol. 2610. Springer, 238–253.

SU, S., ZHANG, C., AND CHEN, J. 2007. An improved genetic algorithm for web services selection. In *Proceedings of the 7th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS'07)*. Lecture Notes in Computer Science, vol. 4531. Springer, 284–295.

SUTTON, A., KAGDI, H., MALETIC, J. I., AND VOLKERT, L. G. 2005. Hybridizing evolutionary algorithms and clustering algorithms to find source-code clones. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'05)*. ACM Press, New York, 1079–1080.

TLILI, M., WAPPLER, S., AND STHAMER, H. 2006. Improving evolutionary real-time testing. In *Proceed-ings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*. ACM Press, New York, 1917–1924.

VAN BELLE, T. AND ACKLEY, D. H. 2002. Code factoring and the evolution of evolvability. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'02)*. Morgan Kaufmann, 1383–1390.

VAN DEN AKKER, M., BRINKKEMPER, S., DIEPLEN, G., AND VERSENDAAL, J. 2005. Flexible release planning using integer linear programming. In *Proceedings of the 11th International Workshop on Requirements Engineering for Software Quality (RefsQ'05)*. 247–262.

VIJAYALAKSHMI, K., RAMARAJ, N., AND AMUT HAKKANNAN, R. 2008. Improvement of component selection process using genetic algorithm for component-based software development. *Int. J. Inf. Syst. Change Manag. 3*, 1, 63–80.

VIVANCO, R. AND JIN, D. 2008. Enhancing preditive models using principal component analysis and search based metric selection: A comparative study. In *Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'08)*. ACM Press, New York, 273–275.

VIVANCO, R. AND PIZZI, N. 2004. Finding effective software metrics to classify maintainability using a parallel genetic algorithm. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'04)*. Lecture Notes in Computer Science, vol. 3103. Springer, 1388–1399.

VIVANCO, R. A. AND JIN, D. 2007. Selecting object-oriented source code metrics to improve predictive models using a parallel genetic algorithm. In *Proceedings of Conference on Object Oriented Programming Systems Languages and Applications the 22nd ACM SIGPLAN Conference on Object-Oriented Programming Systems and Applications Companion (OOPSLA'07)*. ACM Press, New York, 769–770.

WANG, Z., TANG, K., AND YAO, X. 2008. A multi-objective approach to testing resource allocation in modular software systems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'08)*. IEEE, 1148–1153.

WEGENER, J., BARESEL, A., AND STHAMER, H. 2001. Evolutionary test environment for automatic struc-tural testing. *Inf. Softw. Technol. 43*, 14, 841–854.

WEGENER, J. AND GROCHTMANN, M. 1998. Verifying timing constraints of real-time systems by means of evolutionary testing. *Real-Time Syst. 15*, 3, 275–298.

WEGENER, J., GROCHTMANN, M., AND JONES, B. 1997a. Testing temporal correctness of real-time systems by means of genetic algorithms. In *Proceedings of the 10th International Software Quality Week (QW'97).*

WEGENER, J. AND MUELLER, F. 2001. A comparison of static analysis and evolutionary testing for the verification of timing constraints. *Real-Time Syst. 21*, 3, 241–268.

WEGENER, J., STHAMER, H., JONES, B. F., AND EYRES, D. E. 1997b. Testing real-time systems using genetic algorithms. *Softw. Qual. J. 6*, 2, 127–135.

WEN, F. AND LIN, C.-M. 2008. Multistage human resource allocation for software development by multi objective genetic algorithm. *The Open Appl. Math. J. 2*, 95–103.

WHITE, D. R., CLARK, J. A., JACOB, J., AND POULDING, S. M. 2008. Searching for resource-efficient programs: Low-Power pseudorandom number generators. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO'08),* M. Keijzer, Ed., ACM Press, New York, 1775–1782.

WILLIAMS, K. P. 1998. Evolutionary algorithms for automatic parallelization. Ph.D. thesis, University of Reading, UK.

WINDISCH, A., WAPPLER, S., AND WEGENER, J. 2007. Applying particle swarm optimization to software testing. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07).* ACM Press, New York, 1121–1128.

XIE, T., TILLMANN, N., DE HALLEUX, P., AND SCHULTE, W. 2008. Fitness-Guided path exploration in dynamic symbolic execution. Tech. rep. MSR-TR-2008-123, Microsoft Research. September.

YANG, L., JONES, B. F., AND YANG, S.-H. 2006. Genetic algorithm based software integration with minimum software risk. *Inf. Softw. Technol. 48*, 3, 133–141.

YOO, S. AND HARMAN, M. 2011. Regression testing minimisation, selection and prioritisation: A survey. *J. Softw. Test. Verif. Reliab.* (To appear).

YOO, S., HARMAN, M., AND UR, S. 2009. Measuring and improving latency to avoid test suite wear out. In *Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation Workshops (ICSTW'09).* IEEE, 101–110 (Best Paper Award).

YOO, S., HARMAN, M., AND UR, S. 2011a. Highly scalable multi-objective test suite minimisation using graphics card. Tech. rep. RN/11/07, University College London. January.

YOO, S., HARMAN, M., AND UR, S. 2011b. Highly scalable multi-objective test suite minimisation using graphics cards. In *Proceedings of the 3rd International Symposium on Search Based Software Engineering (SSBSE'11).* Springer.

ZELLER, A. 2011. Search-Based program analysis. In *Proceedings of the 3rd International Symposium on Search Based Software Engineering (SSBSE'11).* Springer, 1–4. Keynote.

ZHANG, C., SU, S., AND CHEN, J. 2006. A novel genetic algorithm for QoS-aware web services selection. In *Proceedings of the 2nd International Workshop on Data Engineering Issues in E-Commerce and Services (DEECS'06).* Lecture Notes in Computer Science, vol. 4055. Springer, 224–235.

ZHANG, C., SU, S., AND CHEN, J. 2007a. DiGA: Population diversity handling genetic algorithm for QoS-aware web services selection. *Comput. Comm. 30,* 5, 1082–1090.

ZHANG, X., MENG, H., AND JIAO, L. 2005. Intelligent particle swarm optimization in multiobjective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*. Vol. 1. IEEE, 714–719.

ZHANG, Y., ALBA, E., DURILLO, J. J., ELDH, S., AND HARMAN, M. 2010. Today/Future importance analysis. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO'10).* ACM Press, New York, 1357–1364.

ZHANG, Y., FINKELSTEIN, A., AND HARMAN, M. 2008. Search based requirements optimisation: Existing work and challenges. In *Proceedings of the 14th International Working Conference, Requirements Engineering: Foundation for Software Quality (RefsQ'08).* Lecture Notes in Computer Science, vol. 5025. Springer, 88–94.

ZHANG, Y., HARMAN, M., AND MANSOURI, S. A. 2007b. The multi-objective next release problem. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07).* ACM Press, New York, 1129–1137.

Received April 2009; revised February 2011; accepted August 2011